HELSINKI UNIVERSITY OF TECHNOLOGY
Department of Electrical and Communications Engineering

Functional Data Dimensionality Reduction for Machine Learning

Master's Thesis

Tuomas Kärnä

Laboratory of Computer and Information Science
Espoo 2007

HELSINKI UNIVERSITY OF TECHNOLOGY
Department of Electrical and Communications Engineering

ABSTRACT OF MASTER'S THESIS

Author: Tuomas Kärnä

Title of thesis:
Functional Data Dimensionality Reduction for Machine Learning

Date: October 30th 2007 **Pages:** 10 + 52
Professorship: Computer and Information Science **Code:** T-115
Supervisor: Professor Olli Simula
Instructor: Docent Amaury Lendasse

High dimensional data are becoming more and more common in the field of multivariate data analysis. However, the high dimensionality is problematic due to increasing computational costs and to the curse of dimensionality.

This thesis concerns dimensionality reduction method that is based on Functional Data Analysis. High dimensional data are projected on a function space where it can be expressed in more compact form. The functions space is defined by a set of Gaussian basis functions that are specially adjusted to suit the problem at hand.

The methodology is tested in two applications, chemometrics and time series prediction, using Least-Squares Support Vector Machines for regression. The experiential results indicate that data dimension can be dramatically reduced. And what is more, the prediction accuracy is clearly better or at least equivalent compared to other commonly used methods.

Keywords: Dimensionality reduction, Functional Data Analysis, Chemometrics, Time series prediction
Language: English

TEKNILLINEN KORKEAKOULU DIPLOMITYÖN TIIVISTELMÄ
Sähkö- ja tietoliikennetekniikan osasto

Tekijä:	Tuomas Kärnä	
Työn nimi:	Funktionalinen dimensionalisuuden pienentäminen koneoppimista varten	
Päiväys:	30. syyskuuta 2004	**Sivumäärä:** 10 + 52
Professuuri:	Informaatiotekniikka	**Koodi:** T-115
Työn valvoja:	Professori Olli Simula	
Työn ohjaaja:	Dosentti Amaury Lendasse	

Monimuuttuja-analyysissä korkeadimensioinen informaatio on yleistyy jatkuvasti. Korkean dimension seurauksena laskenta-ajat kasvavat ja ongelmia aiheutuu myös nk. dimensionalisuuden kirouksen (curse of dimensionality) seurauksena.

Tämä diplomityö koskee funktionaliseen data analyysiin perustuvaa dimensionalisuuden pienetämismenetelmää. Tässä menetelmässä korkeadimensioinen informaatio projisoidaan funktioavaruuteen jossa se voidaan kuvata yksinkertasemmassa muodossa. Funktioavaruus määritellään Gaussisten kantafunktioiden avulla, jotka on sovitetty kyseessä olevaan ongelmaan mahdollisimman hyvin.

Esitetyttyä menetelmää sovelletaan kemometriaan ja aikasarjaennustukseen. Regressioon käytetään molemmissa tapauksissa pienimmän neliösumman tukivektorikonetta (Least-Squares Support Vector Machine). Koetulokset osoittavat, että dimensionalisuutta voidaan pienentää merkittävästi. Lisäksi saavutettu ennustustarkkuus on parempi tai vähintään samantasoinen verrattuna muihin yleisesti käytössä oleviin menetelmiin.

Avainsanat:	dimension pienentäminen, funktionaalinen data analyysi, kemometria, aikasarjaennustus
Kieli:	englanti

Acknowledgements

The thesis is written in the Laboratory of Computer and Information Science in Helsinki University of Technology. First of all I would like to express my gratitude to Professor Olli Simula who has been supervising the research. I would also like to thank my instructor Docent Amaury Lendasse for all the discussions and suggestions that have been valuable for doing the experiments and writing the thesis. I am grateful for the opportunity I have had to work in such an encouraging and well-equipped research laboratory as CIS. I would also like to thank all the members of time series and chemoinformatics group and other colleaques of the laboratory, Francesco Corona, Patric Bas, Elia Liitiäinen, Yoan Miche, Antti Sorjamaa, Yu Qi and Emil Eirola for the recent years.

Espoo October 30th 2007

Tuomas Kärnä

Abbreviations and Acronyms

FB	Forward-Backward selection
FDA	Functional Data Analysis
k-NN	k-Nearest Neighbour
KKT	Karush-Kuhn-Tucker
LOO	Leave-One-Out
LS-SVM	Least Squares Support Vector Machine
MSE	Mean Square Error
NMSE	Normalised Mean Square Error
PCR	Principal Components Regression
PLS	Partial Least Squares
RMSE	Root Mean-Square Error
SEP	Standard Error of Prediction
SST	Sea Surface Temperature
SVM	Support Vector Machine

Contents

List of Tables

List of Figures

Chapter 1

Intoduction

1.1 Scope of the thesis

In the field of multivariate data analysis high dimensional data are becoming more and more common. However, in the machine learning perspective, the constantly growing data dimension causes severe problems. First of all, computational complexity of many commonly used analysis methods, such as variable selection, grows exponentially with respect to the number of variables [1]. But what is more important, the analysis suffers from the curse of dimensionality, which states that the theoretical lower bound of error increases with data dimensionality. For example, it has been shown that inference based on pairwise distances becomes increasingly difficult as the dimension of the space grows [2, 3]. Still, most of the commonly used prediction methods, such as k-Nearest-Neighbour (k-NN) [4] and most of kernel methods such as Radial Basis Function Networks (RBFN [1]), Support Vector Machines (SVM [1]) and Kernel Partial Least Squares (K-PLS [5]), rely on pairwise distances and are thus bound to suffer from the curse of dimensionality [6].

The growth of data dimension also implies that more training examples are needed for building a reliable prediction model [2, 6]. However, such requirements are rarely met in practical machine learning applications. Actually, in some cases the number of variables may exceed the number of training examples which is a poor starting point for machine learning and it very likely leads to poor generalisation performance.

To overcome the curse of dimensionality, one can focus on studying only a small subset of the data or project the data into a smaller dimensional

space. Although the first alternative is often effective, it is not efficient: finding a relevant subset can be very time consuming. On the other hand, projecting the data on a small dimensional space often provides a straight forward technique for dimensionality reduction. It should be noted, however, that achieving lossless dimensionality reduction is possible only when the intrinsic dimension of the data is small, i.e. the data lies in a some bounded manifold in a high dimensional space.

In literature there are many non-linear projection tools that attempt to find projections in such a way that the topological structure of the data is preserved as much as possible. Examples of such methods are Curvilinear Component Analysis (CCA) [7], Isomap [8] and Laplacian Eigenmap [9].

This thesis, however, concentrates on a dimensionality reduction that is inspired by Functional Data Analysis (FDA) [10]. The FDA approach is based on the assumption that multivariate data are discrete samples of some underlying functions. The functions are approximated using standard regression techniques, for example, and further analysis is carried out with the functional representations instead of the original data. The advantage is that, provided that the data is smooth enough, the functional representation will be of smaller dimension.

Although there are many ways for obtaining a functional representation, the most commonly utilised one is to construct a finite dimensional function space and to project the data onto this space. The function space is constructed by choosing a set of basis functions a priori. However, the choice is not trivial because in many real-life applications the basis has a severe impact on the overall performance. This thesis presents a functional dimension reduction method that employs problem specific basis functions. The basis is constructed by taking an initial set of Gaussian functions and optimising the locations and widths to achieve a better fitting quality. Thus a good representation can be achieved with small number of basis functions while maintaining most of the original information.

Compared to the non-linear projection methods, the functional dimensionality reduction method is more restricted because one has to assume that the multidimensional data can be represented in functional form. Nevertheless, many real-life applications do involve data that can be regarded as sampled functions. This thesis covers two of such examples: an application to spectral data and time series prediction.

1.2 Publications

This thesis is based on the following publications presented in chronological order.

Publication [11] discusses the utilisation of Gaussian fitting in predicting irregularly sampled time series using both k-NN and Least-Squares Support Vector Machine (LS-SVM) models. In this paper, fixed Gaussian basis functions are used. The functions share one global width parameter that is validated during the training process. The methodology is experimented on a regularly sampled data where one third of the data points have been artificially removed.

Publication [12] a compares B-splines, wavelets and Gaussian functions in FDA based time series prediction task. The experiments were carried out using the ESTSP'07 benchmark data and k-NN predictor. The time series is divided into input and output windows of equal length, and the functional dimensionality reduction is applied to both of them. This paper also introduces the Quasi-Newton optimised Gaussian basis which is compared to a fixed Gaussian basis that employs one global width parameter.

In [13] the Quasi-Newton optimised basis is applied to chemometrics. Spectral data is compressed using the function approximation and a LS-SVM model is trained to predict certain chemical quantities based on the spectrometric information. Experimental results obtained with two data sets from the food industry are presented.

Publication [14] is an extended version of the latter conference paper. The methodology remains essentially the same, but experiments are carried out with one more data set and the results are compared to other commonly-used methods in chemometrics, namely Partial Least Squares (PLS) and Principal Component Regression (PCR). In addition, experiments with variable selection are also presented.

Publication [15] discusses the use of functional dimension reduction in long-term time series prediction. The methodology is a generalisation to that presented in [12]: The time series is again divided into input and output windows, but in contrast to [12], the windows do not have to be of equal length. The Gaussian basis functions are optimised separately for the inputs and the outputs. Two real-world applications are presented: Prediction of climatological time series and electricity consumption forecasting.

1.3 Structure of the thesis

The thesis is organised as follows. Chapter 2 presents the methodology starting from a general regression problem, discussing the related common techniques and the LS-SVM and k-NN regression models. The latter half of the chapter introduces the functional dimensionality reduction as well as the Quasi-Newton optimisation of the basis functions. The next two chapters discusses the applications to real-world problems: Application to chemometrics is presented in Chapter 3 and an application to long-term time series prediction is presented in Chapter 4. Finally, conclusive remarks are given in Chapter 5.

Chapter 2

Functional data dimensionality reduction

2.1 General regression problem

Let $(\mathbf{z}_i, p_i)_{i=1}^n \in \mathbb{R}^m \times \mathbb{R}$ be a set of n input-output pairs. The classical regression problem is to find a relation that explains the *responce variable* p_i as precisely as possible using the *explanatory variables* $\boldsymbol{z}_i$. Expressed in mathematical terms, the goal is to find a *model* f for which it holds $p_i = f(\boldsymbol{z}_i) + e_i$ so that the error (sometimes called noise or residual) e_i is as small as possible. Traditionally f is a linear model, since many phenomena, especially in engineering, are or can be assumed to be linear. This work, however, focuses mainly on non-linear models.

In many real life cases it is not obvious a priori which model should be chosen for a task. Moreover, the model usually involves some parameters that need to be tuned in order to minimise the error. In statistics the parameter tuning is often called *fitting* while in machine learning more human-centred words such as *training* or *learning* are used.

In mathematics, the problem setting where inputs $\boldsymbol{z}_i$ and the responses p_i are known but the model is not, is known as an *inverse problem* [16]. The corresponding forward problem would be to compute the responses using the inputs and the known model. Inverse problems are often difficult to solve due to ill-posedness: there might not be a unique solution or the solution might be unstable, i.e. the model parameters would depend heavily on small perturbations in the observations. A common cure for the ill-posedness is regularisation where the problem is replaced by a similar but almost well-posed

problem. The LS-SVMs, presented in Section 2.2.1, utilises this approach.

The next section discusses good properties and common pitfalls of non-linear regression models in a machine learning perspective.

2.1.1 Generalisation and over fitting

The goal of any model is to be general, i.e. to perform well with new, previously unseen samples. The ability to generalise is related to the complexity of the model: Any given data set can be fitted exactly if the model is complex enough. However, the drawback is that these models are poor interpolants, i.e. they fit the data points well but behave poorly in between. In machine learning the phenomenon is called over fitting or poor generalisation performance [1].

To avoid over fitting, the data set is usually divided in to three distinct data sets: learning set C_L, validation set C_V and test set C_T. The regression model is trained in the learning set and the parameter values are chosen according to the results in the validation set. Finally, the obtained model is simulated on the test set. Prediction accuracy on the test set gives a rough estimate of how well the model is able to predict new data. For this purpose it is important to keep the test set independent, i.e. all the parameter tuning and model selection should be based on the performance in the validation (or learning) set.

However, dividing the data set into three parts may be problematic due to limited amount of available data. Therefore, the need for a separate validation set is often circumvented by using re-sampling methods, such as K-fold cross validation [1] or bootstrap [17].

K-fold Cross validation

In K-fold cross validation, the learning set is divided in to K bins of equal size. Each of the bins is used as a validation set and the model is trained using the remaining $K-1$ bins. The final cross validation error is an average of all the K validation errors.

If K equals to the number of training examples, each bin consists of only one example, the process is called have Leave-One-Out (LOO) validation. On the other hand, if $K = 1$, we have the conventional one-set training method. It should be emphasised that none of these choices of K give objective measure of model quality. Rather, the obtained values are simply different estimates

of the prediction performance, although in practice it can be said that large K tends to give more trustworthy results. Naturally, the drawback of of cross validation is that the computational load becomes roughly K-fold.

2.1.2 Curse of dimensionality

In machine learning, the term curse of dimensionality refers to disadvantages that are due to high data dimensionality. There are many definitions in the literature, but generally the curse of dimensionality states that the performance of a learning machine tends to degrade as the input data dimension grows. This is due to the fact that the reliability of the model arises from the density of the training examples in the input space. In other words, in order to reach a certain generalisation error level, the required number of training examples grows exponentially versus the data dimension: If, for instance, 10 examples per unit length is sufficient in one dimension, to achieve same information density in two dimensions one needs 100 points per unit square, in three dimensions 1000 points per unit cube and so forth [2].

Bengio et al. [6] have shown that in the case of general kernel machine, the number of training examples required to learn a specific function is directly proportional to the complexity of the function. The curse of dimensionality arises from the fact that the complexity (such as the number of sign changes, as in their example) can grow exponentially with the data dimension. This suggest that in order to cope with the curse of dimensionality, the target function must become smoother as the data dimension grows [1]. In other words, learning complex phenomena gets increasingly difficult as the data dimension crows.

Another aspect to the curse of dimensionality is the concentration of pairwise distances. It has been shown that pairwise distances of random vectors in $\mathbb{R}^n$ tend to concentrate on a small interval in a high dimensional space. As the dimension grows the interval becomes narrower and moves further away from zero [2]. This implies that the distances to an arbitrary query point tend to become equally long and thus the consept of nearest neighbour becomes unclear. Obviously, the same phenomenon affects kernel machines as well: The output of kernel function also concentrates on a narrow range and the function is therefore unable to distinguish the input patters clearly.

2.2 Regression models

2.2.1 LS-SVM

Least Squares Support Vector Machine (LS-SVM) is a least square modification of the Support Vector Machine (SVM) that was introduced by Suykens [18]. LS-SVM has two advantages over SVM: First, the computationally demanding quadratic optimisation problem of SVM is simplified so that it reduces to a set of linear equations which greatly decreases the computational costs. Secondly, regression SVM involves three unknown parameters while LS-SVM has only two, the regularisation parameter γ and the kernel width σ, which significantly simplifies the parameter optimisation phase. Despite these simplifications, LS-SVM still has the most important property of SVM, the absence of local minima in the parameter optimisation.

Consider the set of n input-output pairs $(\mathbf{z}_i, p_i)_{i=1}^n \in \mathbb{R}^m \times \mathbb{R}$. The LS-SVM model is $\hat{p} = \mathbf{w}^T \boldsymbol{\psi}(\mathbf{z}) + b$, where $\boldsymbol{\psi} : \mathbb{R}^m \longmapsto \mathbb{R}^l$ is a mapping from the input space onto a higher dimensional hidden space, $\mathbf{w} \in \mathbb{R}^l$ is a weight vector and b is a bias term. The optimisation problem is formulated as

$$\begin{aligned} \min_{\mathbf{w},b} \; J(\mathbf{w}, e) &= \frac{1}{2}\|\mathbf{w}\|^2 + \frac{1}{2}\gamma \sum_{i=1}^{n} e_i^2, \\ \text{so that} \quad p_i &= \mathbf{w}^T \boldsymbol{\psi}(\mathbf{z}_i) + b + e_i, \end{aligned} \tag{2.1}$$

where e_i is the prediction error. The regularisation parameter $\gamma \geq 0$ controls the trade-off between flatness of the function and accuracy of the function. The dual problem is derived using Lagrangian multipliers which leads into a linear KKT system [18]. Using the dual solution, the original model can be reformulated as

$$\hat{y} = \sum_{i=1}^{n} \alpha_i K(\mathbf{z}, \mathbf{z}_i) + b, \tag{2.2}$$

where the kernel $K(\mathbf{z}, \mathbf{z_i}) = \boldsymbol{\psi}(\mathbf{z})^T \boldsymbol{\psi}(\mathbf{z}_i)$ is a continuous and symmetric mapping from $\mathbb{R}^m \times \mathbb{R}^m$ to $\mathbb{R}$ and α_i are the Lagrange multipliers. It should be emphasised that although we formally define the high dimensional hidden space $\mathbb{R}^l$ and the mapping $\boldsymbol{\psi}(\mathbf{z})$, there is no need to compute anything in the hidden space; The knowledge of the kernel K is enough. A widely-used choice is the standard Gaussian kernel

$$K(\mathbf{z}_1, \mathbf{z}_2) = e^{-\|\mathbf{z}_1 - \mathbf{z}_2\|_2^2 / \sigma^2}. \tag{2.3}$$

2.2.2 k-NN

In pattern recognition, k-nearest neighbour (k-NN) algorithm has been used for clustering or classification [4], but it is also suitable for regression. For example, Sorjamaa et al. [19] have presented an application to time series prediction.

Consider again the training examples $(\mathbf{z}_i, p_i)_{i=1}^n \in \mathbb{R}^m \times \mathbb{R}$. With k-NN, the estimate of the output is obtained as a mean of the k most similar outputs. I.e. given a previously unknown input $\mathbf{z}$ we get the estimate,

$$\hat{p} = \frac{1}{k} \sum_{\{p_i | \mathbf{z}_i \in \mathcal{N}_k(\mathbf{z})\}} p_i, \tag{2.4}$$

where the set $\mathcal{N}_k(\mathbf{z})$ is the k nearest neighbors of $\mathbf{z}$. Usually the Euclidian metric is used to compute the distance between samples, but other metrics can be used as well. The number of neighbours, k, is unknown and need to be validated separately. Clearly, the output could be replaced with a p_i in $\mathbb{R}^q$ without any changes in the formulation. Thus, in contrast to LS-SVM, k-NN is cabable of modelling multidimensional outputs directly.

Because k-NN is computationally very cheap method, it is convenient in time-consuming tasks such as variable selection or exhaustive parameter tuning [12, 19]. Other advantages are that k-NN is quite robust method but is still able to model very nonlinear phenomena. However, k-NN suffers from the curse of dimensionality: Since k-NN computes averages of the training example outputs, the performance depends heavily on the density of the examples. Therefore k-NN does not usually perform very well on small data sets. Moreover, in high dimensional spaces the concept of nearest neighbors is vague [2, 3].

2.3 Functional dimensionality reduction

The central idea of Functional Data Analysis (FDA) is to treat multivariate data as continuous functions [10]. Applied to the regression problem at hand, the inputs $\boldsymbol{z}_i = [z_i^1, z_i^2, \ldots, z_i^m]^T$ are assumed to be (possibly noisy) measurements of some underlying continuous function $v_i(x)$. To be more specific, it is assumed that for each i there exists a function $v_i(x)$ so that $z_i^h = v_i(x_h) + \epsilon_i^h$ for all $h = 1, \ldots, m$, where ϵ_i^h stands for the observation noise. The arguments x_h where the observations are made are problem specific, for example in the case of spectral data, it would be natural that x_h corresponds

to the the wavelengths of measurements. Generally speaking, however, x_h can be chosen to be any increasing indexing, such as $x_h = h,\ h = 1, \ldots, m$.

Without much loss of generality it can be assumed that the function v_i belongs to $L^2[a, b]$, the space of square integrable functions on some interval $[a, b]$ sufficiently large. The idea is to estimate the underlying functions v_i based on the available data and conduct further analysis with the function representation instead of the original discrete data. In other words, the original regression problem becomes $p_i = f(v_i) + e_i$.

However, the space $L^2[a, b]$, like function spaces in general, is infinite dimensional which implies that it is impossible to work with the functions v_i in practice. To circumvent the problem, v_i is approximated with a finite dimensional representation $\boldsymbol{\omega}_i$. Since the goal of this whole procedure is to reduce dimensionality, a natural requirement is that $\dim(\boldsymbol{\omega}_i) < \dim(\boldsymbol{z}_i)$. Using the representation $\boldsymbol{\omega}_i$, the regression problem can be written in a practically implementable form: $p_i = f(\boldsymbol{\omega}_i) + e_i$.

2.3.1 Finite dimensional representation of functions

In order to obtain a finite dimensional representation of v_i, it is necessary to consider some q dimensional subspace $\mathcal{A} \subset L^2[a, b]$. The subspace is defined by a set of basis functions $\varphi_j(x),\ j = 1, \ldots, q$ which spans $\mathcal{A}$. In fact, $\mathcal{A}$ is a normed vector space and a natural choice for the norm is that of the $L^2[a, b]$ space: $\| v \|_{\mathcal{A}} = \left(\int_a^b v(x)^2 dx \right)^{1/2}$

Given the basis, any function v_i in $\mathcal{A}$ can be uniquely defined by a weight vector $\boldsymbol{\omega}_i = [\omega_i^1, \omega_i^2, \ldots, \omega_i^q]^T$:

$$v_i(x) = \sum_{j=1}^{q} \omega_i^j \varphi_j(x) = \boldsymbol{\omega}_i^T \boldsymbol{\varphi}(x), \tag{2.5}$$

where $\boldsymbol{\varphi}(x) = [\varphi_1(x), \varphi_2(x), \ldots, \varphi_q(x)]^T$. The weights are obtained by minimising the square fitting error[1] :

$$E_i(\boldsymbol{\omega}_i) = \sum_{h=1}^{m} (\boldsymbol{\omega}_i^T \boldsymbol{\varphi}(x_h) - z_i^h)^2. \tag{2.6}$$

[1] Obviously, the function fitting is also a regression task, except in one dimension, and most of what has been said about regression is applicable. However in this work, for clarity, the word regression is preserved for the original multivariate regression task, while computing the functional approximation is referred to as fitting.

The solution is the pseudo-inverse $\boldsymbol{\omega}_i = (\mathbf{G}^T\mathbf{G})^{-1}\mathbf{G}^T\mathbf{z}_i$, where the matrix elements are given by $[\mathbf{G}]_{h,j} = \varphi_j(x_h)$ and $\mathbf{G} \in \mathbb{R}^{m\times q}$ [1]. In practice, one may use regularisation to reduce instability of the fitting. In conventional regularisation, a penalty term $\tau\|\boldsymbol{\omega}_i\|^2$ is added to the cost function 2.6. Therefore, the optimisation favours small weights and thus results in a more stable model. The importance of the penalisation is controlled by the regularisation parameter $\tau > 0$. In this case, the solution becomes $\boldsymbol{\omega}_i = (\mathbf{G}^T\mathbf{G}+\tau\mathbf{I})^{-1}\mathbf{G}^T\mathbf{z}_i$ [1].

Now, any function $v_i \in \mathcal{A}$ has an unique representation $\boldsymbol{\omega}_i$ and vice versa, which suggests that in order to compare two functions, it is sufficient to compare the weight vectors instead. However, it should be noted that the functional fitting distorts distances, which can be seen by computing the distance of two arbitrary functions $v(x) = \boldsymbol{\omega}^T\boldsymbol{\varphi}(x)$ and $u(x) = \boldsymbol{\xi}^T\boldsymbol{\varphi}(x)$ in the function space:

$$\begin{aligned} \| v - u \|_{\mathcal{A}}^2 &= \int_a^b \Big((\boldsymbol{\omega} - \boldsymbol{\xi})^T\boldsymbol{\varphi}(x) \Big)^2 dx = (\boldsymbol{\omega} - \boldsymbol{\xi})^T\boldsymbol{\Phi}(\boldsymbol{\omega} - \boldsymbol{\xi}) \qquad (2.7) \\ \boldsymbol{\Phi}_{i,j} &= \int_a^b \varphi_i(x)\varphi_j(x)dx \end{aligned}$$

Clearly, if the functions are orthonormal, $\boldsymbol{\Phi}$ becomes an identity and the norm is equal to the norm in (Euclidian) weight space, $(\boldsymbol{\omega} - \boldsymbol{\xi})^T(\boldsymbol{\omega} - \boldsymbol{\xi})$. Generally this is not the case and distances in function space are not equivalent to those in the weight space. The distortion can be circumvented by applying a linear transformation $\tilde{\boldsymbol{\omega}} = \mathbf{U}\boldsymbol{\omega}$, where $\mathbf{U}$ is the Cholesky decomposition of $\boldsymbol{\Phi} = \mathbf{U}^T\mathbf{U}$ [20]. This method has been used in publications [11, 12, 13].

In FDA, standard basis functions, such as Fourier basis, B-splines [21] or wavelets [22] are often used [10, 12]. However, instead of using a basis that is fixed a priori, it is appealing to build a problem specific basis so that minimal number of weights are needed for representing the data.

A qualitative comparison of different basis functions is presented in Table 2.1. The Fourier basis functions and the wavelets are orthonormal, but there are no free parameters and thus the basis cannot be optimised for a specific task. In the case of B-splines, on the other hand, one can choose the break-points of the piecewise polynomials. Nonetheless there are rather strict restrictions for these break-points [21], which would cause problems in the optimisation process.

In this work, Gaussian basis functions are suggested for the dimension reduction. The basis functions are given by:

$$\varphi_j(x) = e^{-\|x-r_j\|^2/s_j^2}, \; j = 1, \ldots, q, \qquad (2.8)$$

Table 2.1: Comparison of common basis functions. Fourier and wavelet basis are orthogonal, but cannot be tuned to fit specific data. Among the functions presented here, only Gaussians are suitable for unconstrained optimisation.

Basis	Ortho-gonal	Local functions	Adjustable parameters	Unconstrained parameters
Fourier	√			
Wavelet	√	√		
B-splines		√	√	
Gaussian		√	√	√

where r_j defines the location and s_j the width of the function. Clearly, these functions are not orthonormal, but there are no constraints for the parameters r_j and s_j. Moreover, the basis is easily differentiable and thus it can be optimised with any non-linear unconstrained optimisation algorithm.

2.3.2 Optimising the Gaussian basis functions

Gaussian functions are differentiable with respect to the parameters r_j and s_j, and thus the basis can be optimised for an accurate fitting using standard gradient-based methods.

To derive the gradient, we first define the error functional. Squared fitting error of all the functions $i = 1, \ldots, n$ can be written as,

$$\begin{aligned} E &= \frac{1}{2}\sum_{i=1}^{n} (\mathbf{G}\boldsymbol{\omega}_i - \mathbf{z}_i)^T(\mathbf{G}\boldsymbol{\omega}_i - \mathbf{z}_i) \\ &= \frac{1}{2}\sum_{i=1}^{n} (\boldsymbol{\omega}_i^T\mathbf{G}^T\mathbf{G}\boldsymbol{\omega}_i - 2\mathbf{z}_i^T\mathbf{G}\boldsymbol{\omega}_i + \mathbf{z}_i^T\mathbf{z}_i) \end{aligned}$$

The columns of $\mathbf{G}$ are denoted as $\mathbf{G}_j = [\varphi_j(x_1), \varphi_j(x_2), \ldots, \varphi_j(x_m)]^T$ and it's derivative with respect to r_j and s_j

$$\begin{aligned} \mathbf{G}_j^{(r)} &= \Big[\frac{x_1 - r_j}{s_j^2}\varphi_j(x_1), \ldots, \frac{x_m - r_j}{s_j^2}\varphi_j(x_m)\Big]^T \\ \mathbf{G}_j^{(s)} &= \Big[\frac{(x_1 - r_j)^2}{s_j^3}\varphi_j(x_1), \ldots, \frac{(x_m - r_j)^2}{s_j^3}\varphi_j(x_m)\Big]^T, \end{aligned}$$

respectively. With this notation we obtain,

$$\begin{aligned}\frac{\partial}{\partial r_j}(\mathbf{z}_i^T\mathbf{G}\boldsymbol{\omega}_i) &= \mathbf{z}_i^T\mathbf{G}_j^{(r)}\omega_i^j \\ \frac{\partial}{\partial r_j}(\boldsymbol{\omega}_i^T\mathbf{G}^T\mathbf{G}\boldsymbol{\omega}_i) &= 2\boldsymbol{\omega}_i^T\mathbf{G}^T\mathbf{G}_j^{(r)}\omega_i^j,\end{aligned}$$

which finally yields,

$$\frac{\partial E}{\partial r_j} = \sum_{i=1}^{n}(\mathbf{G}\boldsymbol{\omega}_i - \mathbf{z}_i)^T\mathbf{G}_j^{(r)}\omega_i^j.$$

Following similar steps for $\partial/\partial s_j$ we get,

$$\frac{\partial E}{\partial s_j} = \sum_{i=1}^{n}(\mathbf{G}\boldsymbol{\omega}_i - \mathbf{z}_i)^T\mathbf{G}_j^{(s)}\omega_i^j.$$

When the gradient is known, the locations and the widths are optimised using unconstrained non-linear optimisation. Actually, the problem is constrained to $s > 0$ but because the basis function (2.8) is even with respect to s, the constraint can be relaxed. In this paper a Broyden-Fletcher-Goldfarb-Shanno (BFGS) Quasi-Newton method with line search is used [23]. BFGS method resembles quadratic algorithms, such as the Newton method, in the sense that it assumes that the problem is quadratic, but with BFGS there is no need to actually compute the Hessian matrix at any stage.

Non-linear optimisation requires an initial set of Gaussian functions. Since there are many local minima involved in the optimisation problem, the choice of initialisation is not trivial. In this work, however, for the sake of simplicity, the basis functions are initially distributed evenly on the data interval and the width is set to the distance between neighboring centres.

2.4 Variable selection

Selection of relevant input variables is a important yet difficult task in machine learning. Irrelevant inputs introduce noise to the prediction model which decreases performance. Reducing the number of variables simplifies the model and the parameter optimisation becomes easier. And what is more, variable selection can provide the researcher valuable information about the problem at hand.

The most simple variable selection method is exhaustive search, i.e. trying out all the possible variable combinations. However, exhaustive search among m variables requires 2^{m-1} iterations which quickly becomes impossible as m grows.

2.4.1 Forward-Backward selection

Forward-Backward (FB) selection [24] is a faster algorithm compared to the exhaustive search but there is no guarantee that the optimal set of variables is found.

In FB algorithm, each variable can be in two states: “on”, meaning that it belongs to the set of selected variables or “off” meaning that it is discarded. Given a certain initial state vector (states of all variables), the algorithm proceeds by flipping the state of each variable at a time and by computing the corresponding error measure. The flip operation that improved performance the most is accepted, resulting in a new state vector. Next, the states are flipped again (excluding the previously accepted change). The process is continued until no improvement is found. FB selection can be seen as descent in a graph where neighboring state vectors differ with exactly one state. Such a graph contains many local minima and therefore it is advisable to initialise the process with random state vectors in addition to the ordinary "all on" and "all off" states.

2.5 Error measures

The standard error measure in statistics and machine learning is Mean Square Error (MSE) defined by

$$\mathrm{MSE}_T = \frac{1}{|C_T|} \sum_{i \in C_T} (p_i - \hat{p}_i)^2,$$

where $\hat{p}_i$ represents the estimated values and the number of examples in the test set is denoted by $|C_T|$.

In this work we are primarily using the Normalised Mean Square Error (NMSE) to measure the quality of the prediction. NMSE is obtained by dividing Mean Square Error (MSE) by the variance of the output,

$$\mathrm{NMSE}_T = \frac{1}{|C_T| \ \mathrm{Var}(p)} \sum_{i \in C_T} (p_i - \hat{p}_i)^2$$

$$\mathrm{Var}(p) \quad = \quad \sum_{i \in C_L \cup C_T} (p_i - \bar{p})^2,$$

where and $\bar{p}$ the mean of the output. The subscripts L and T stand for the learning set and the test set, respectively. NMSE is useful when comparing results between different data sets and it bears close relationship to the R square measure, given by $\mathrm{R}^2 = 1 - \mathrm{NMSE}$.

When comparing the obtained results to literature, errors are sometimes reported in Root Mean Square Error (RMSE) as well. RMSE is simply the square root of MSE.

Chapter 3

Application I: Chemometrics

3.1 Introduction

A typical problem in chemometrics deals with predicting some chemical quantity directly from measured spectrum. Due to additivity of absorption spectra, the problem is assumed to be linear and therefore linear models, such as PLS [25] or PCR [26] have been widely used for the prediction task. However, it has been shown that the additivity assumption is not always true and environmental conditions may further introduce non-linearities to the problem [27]. Therefore, in order to be able to deal with general problems, a non-linear method should be used. Non-linear LS-SVMs models have already been applied to chemometrics (see [28], for example).

In chemometrics high dimensional data is very common: Due to development of more accurate spectrometers one can easily obtain spectra of thousands of data points. First of all, the high input dimension is problematic due to the curse of dimension. As mentioned in Section 2.1.2, the curse of dimension implies that in order to obtain a reliable model the number of training examples should increase exponentially compared to the increase in dimension. In chemometrics, however, the data sets tend to be small because collecting samples from the food industry is rather expensive and time consuming. Nowadays it is not uncommon to encounter data sets where there are more variables than training examples. Naturally this is a very poor starting point for any sort of modelling and it stresses the need for reliable dimensionality reduction.

Another problem that the high data dimension causes is the increase in computational time. For example, variable selection is a widely used method in

many applications because it not only simplifies the model but also provides relevant information about which wavelengths are important for the prediction task. The drawback is that computational time grows exponentially compared to the input dimension. Therefore selecting important variables directly on the raw data is often difficult or impossible.

Spectral data are usually rather smooth and low on noise, so function fitting is a convenient tool for compressing the spectra. In literature, function fitting has been used in chemometrics application (with or without dimension reduction). Commonly used bases are B-splines [29] and wavelets [30, 31].

In this chapter, the proposed functional dimension reduction is applied to chemometrics. Experimental results obtained with three data sets originating from the food industry are presented. The goal in all the cases is to predict some analytical values (such as fat content) from infrared absorption spectra. As explained in Section 2.3, Gaussian fitting is applied to each spectrum and a LS-SVM model is used for the prediction. For comparison, results obtained with PLS [25] and PCR [26] models are also presented. To fully utilise the reduced dimensionality, Forward-Backward (FB) variable selection is used to select relevant basis functions.

The data sets and the experimental setup are presented in Section 3.2. The results are presented in Section 3.3 with some conclusive remarks.

3.2 Experiments

3.2.1 Data sets

The Tecator data set consists of NIR absorption spectra and fat contents of 215 samples of minced pork meat [32, 33]. Each spectrum has been measured at 100 wavelengths ranging from 850nm to 1050nm using Tecator Infratec Food and Feed Analyzer. The fat content ranges from 0.9 to 49.1 per cent. First 172 spectra were used as a learning set C_L and the remaining 43 were used as a test set C_T. Spectra of the learning set are illustrated in Figure 3.1.

The second data set contains 124 mid-infrared absorption spectra of wine samples and the goal is to determine the percentage of alcohol. The 256 spectral variables relate to wavenumbers ranging from 400 to 4000 cm^{-1} [34]. Alcohol content ranges from 7.48 per cent to 18.5 per cent. First 94 spectra were used as a learning set C_L while the remaining 30 were regarded as a test set C_T. The spectra are illustrated in Figure 3.2.

The third data set is related to prediction saccharose concent of orange juice samples. The data set contains absorption spectra of 700 variables measured in range 1000 to 2500 nm [34]. The training and learning set contain 146 and 67 values, respectively. The saccharose content ranges from 0 to 78.8 per cent. The spectral data are illustrated in Figure 3.3.

Both the Wine and Juice data sets are examples of cases where the number of spectral variables exceed the number of training examples.

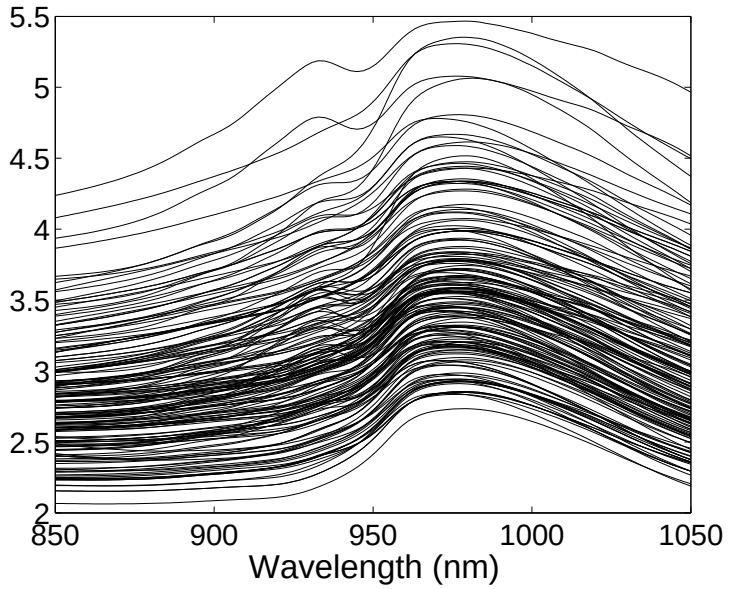

Figure 3.1: Spectra of the Tecator data set.

3.2.2 Selecting the number of basis functions

For each data set, the Gaussian fitting was computed as explained in Section 2.3. The number of Gaussian functions was validated using LS-SVM cross validation. The maximum number of basis functions was 25, 40 and 35 for Tecator, Wine and Juice data sets, respectively. The evolution of fitting accuracy and LS-SVM validation error is presented in Figures 3.4, 3.5 and 3.6 for the three data sets, respectively. Fitting accuracy should decrease monotonically as the number of basis functions increase, but in practice this may not always be true due to local minima in the optimisation process. Ideally, there should be a minimum in the validation error in order to make the selection easy, but this kind of behaviour is clearly visible only with the

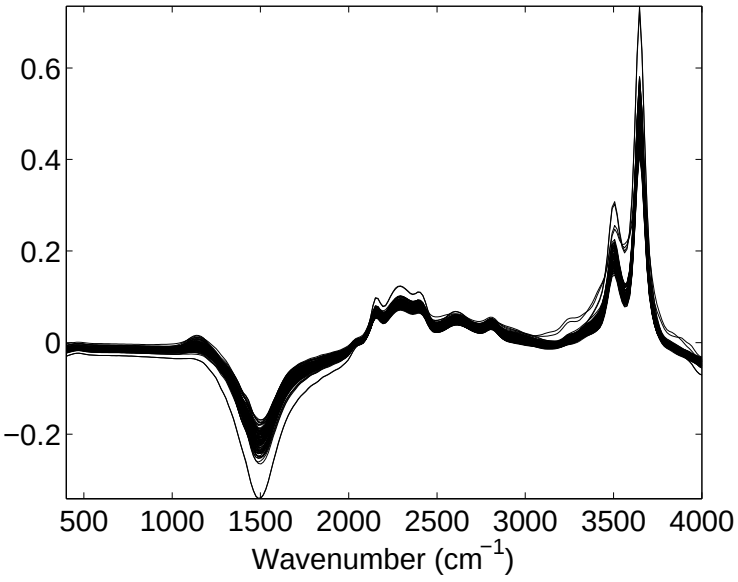

Figure 3.2: Spectra of the Wine data set.

Tecator data. The optimised basis functions and examples of fitted functions are presented in Figures A.1, A.2 and A.3 for the three data sets in the Appendix.

The LS-SVM models were trained using 10-fold cross validation. To acquire reliable cross validation errors, the learning sets were randomly permutated. The same permutation was used in all tests to obtain fully comparable results. The LS-SVM parameters γ and θ were optimised using four sequential 10-by-10 grid searches, starting from a coarse grid and moving to a finer one near the minimum value.

As benchmarks, widely used PLS and PCR regression models were trained on the spectral data. The number of latent variables (in PLS) and number of principal components (in PCA) were selected by LOO validation. The number of basis functions was also selected using LOO.

To illustrate the effect of the Gaussian fitting, all the three models, PCA, PLS and LS-SVM, were trained with the raw spectral data as well. In all cases, the input variables were scaled to zero mean and unit variance before training[1].

[1]To be more specific, the learning set was transformed to zero mean and unit variance and the same transformation was used in the test set

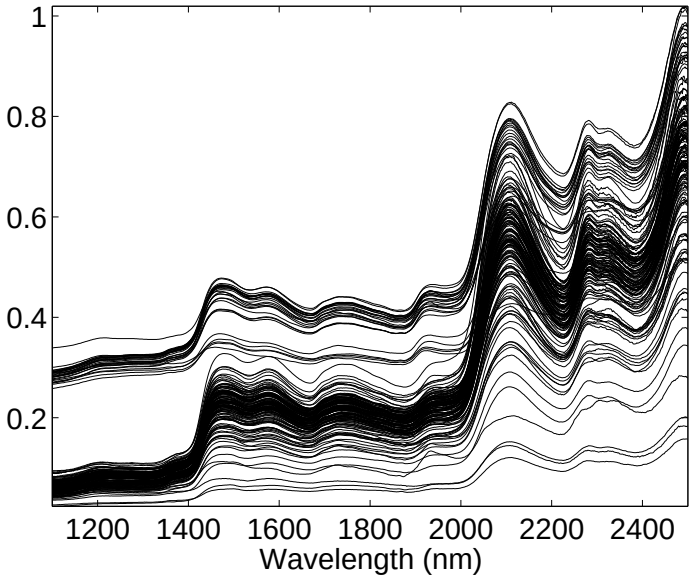

Figure 3.3: Spectra of the Juice data set.

3.2.3 FB variable selection

To initialise FB variable selection, 100 random selections were computed, out of which the 15 best were tuned with the actual FB algorithm.

The variables were chosen based on the LS-SVM validation error. Since the number of variables changes during FB iteration, the LS-SVM parameters, γ and θ, must be adjusted. However, performing a complete grid-search in each step would be computationally too demanding. To speed up the process, the error is computed only in a local 3-by-3 grid $(\gamma, \theta) \in \{1/3\gamma_p, \gamma_p, 3\gamma_p\} \times \{1/3\theta_p, \theta_p, 3\theta_p\}$ where (γ_p, θ_p) is the previous optimum. Clearly, the error estimates are not as accurate as they could be, but for variable selection it is sufficient to obtain a ranking between different sets of variables.

During the selection, a more accurate 13-fold cross validation was used. This was to prevent possible over fitting: 13-fold cross validation provides a fairly reliable error measure while ensuring that the cross validation subsets are different from the final 10-fold cross validation.

After the FB selection, final LS-SVM validation error was computed with full grid search as described in Section 3.2.2. The validation errors after FB selection are plotted in dashed line in Figures 3.4, 3.5 and 3.6. One can

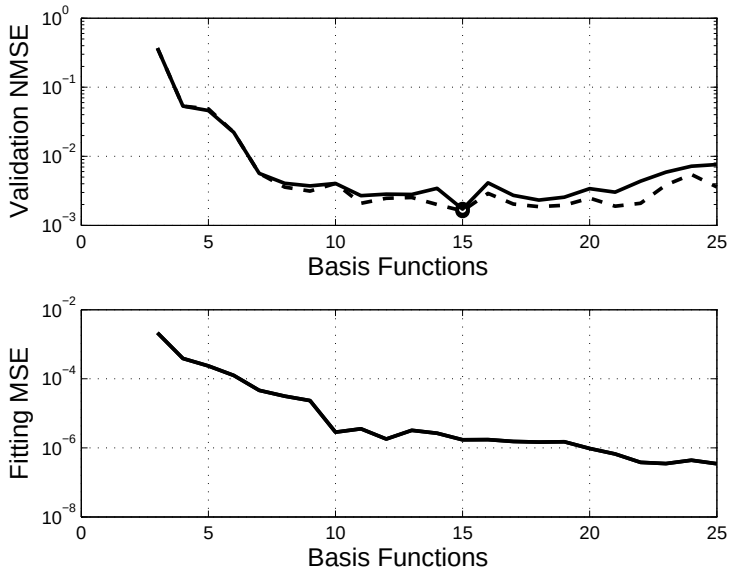

Figure 3.4: Tecator: Validation error and fitting accuracy versus number of basis functions. Prediction performance after FB variable selection is marked with a dashed line. The smallest validation errors are marked with a circle.

observe that the FB selection tends to be more useful as the size of the basis increases.

In the case of the Tecator data set, the smallest validation error was obtained with 15 basis functions out of which 11 were selected. For the other two experiments, most of the variables were discarded selecting only 16 variables out of total 40 in the Wine data set and 11 out of 34 in the Juice data set. Thus the original spectral data was compressed remarkably. The data compression ratios were 9, 16 and 64 for the three data sets, respectively.

3.3 Results

The prediction errors obtained with the three data sets are presented in Tables 3.1, 3.2 and 3.3. In order to ease up comparison to other methods in literature, the test set prediction error is reported in three different measures MSE, NMSE and RMSE. In case of PLS and PCR, the number of latent variables are given in parenthesis.

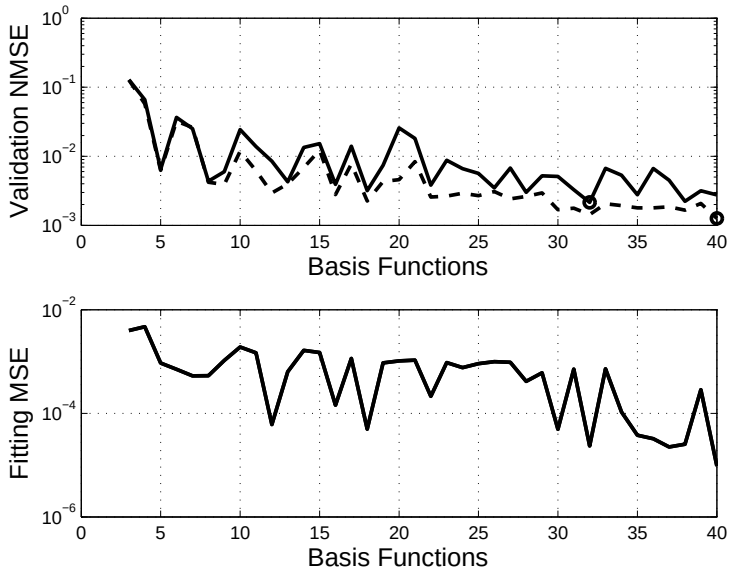

Figure 3.5: Wine: Validation error and fitting accuracy versus number of basis functions. Prediction performance after FB variable selection is marked with a dashed line. The smallest validation errors are marked with a circle.

3.3.1 Tecator

In the case of Tecator data, using non-linear prediction method is clearly advantageous. Using LS-SVM divides the MSE by 5 compared to any of the linear models. This suggests that there are some non-linearities involved in the input-output relation. The non-linearity of this data set has been discussed also in the original work by Borggaard et al. [32].

Using the Gaussian fitting improves the performance even more, roughly by a factor of 4. After variable selection the quality of the prediction is very good (NMSE 0.0012) which can be seen in Figure 3.7 where the actual values are plotted against the predicted ones.

Comparing to other results in literature, Rossi et.al. [24] have reported comparable NMSE 0.0027 obtained with LS-SVM using mutual information based variable selection. In this case, the selection is computed among the original 100 variables.

Thodberg [33] has reported RMSE 0.36 (calling it Standard Error of prediction, SEP) using a committee of Bayesian neural networks. Vila et. al. [35] have reported SEP 0.34 again with Bayesian neural network model with con-

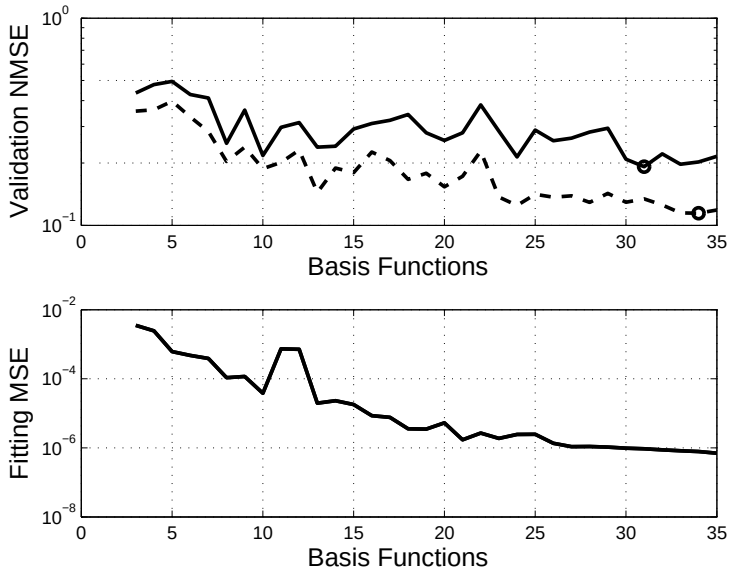

Figure 3.6: Juice: Validation error and fitting accuracy versus number of basis functions. Prediction performance after FB variable selection is marked with a dashed line. The smallest validation errors are marked with a circle.

jugate prior based computation of posterior densities. The smallest RMSE obtained with the proposed Gaussian fitting and FB selection is only slightly worse, 0.43.

3.3.2 Wine

In the case of Wine data set, all the models, including plain PLS and PCR, give good results which suggests that the problem is highly linear. When combined with the Gaussian fitting, LS-SVM performs the best, although PCR is not much worse. After FB selection the number of variables has been reduced down to 16 while the prediction MSE is reduced by roughly 25 per cent. Comparing the predicted values to corresponding targets (Figure 3.7) one can observe that the model is very accurate.

Benoudjit et. al. [34] used several linear methods and Radial Basis Function Networks (RBFN) to address the same problem. The smallest NMSE was 0.0009 that was obtained with a RBFN with FB variable selection. However, the results are not entirely comparable, because Benoudjit et.al. reported validation error and did not use an independent test set.

Table 3.1: Results of the Tecator data set. The number of latent variables are given in parenthesis.

Method	Number of variables	MSE_T	$NMSE_T$	$RMSE_T$
PLS	100 (13)	4.40	0.0271	2.10
PCR	100 (20)	5.00	0.0308	2.24
Fitting + PLS	23 (12)	5.29	0.0325	2.30
Fitting + PCR	23 (17)	5.41	0.0333	2.33
LS-SVM	100	0.99	0.0061	0.99
Fitting + LS-SVM	15	0.28	0.0017	0.53
Fitting + LS-SVM + FB	11	0.19	0.0012	0.43

Table 3.2: Results of the Wine data set. The number of latent variables are given in parenthesis.

Method	Number of variables	MSE_T	$NMSE_T$	$RMSE_T$
PLS	256 (10)	0.0088	0.0043	0.094
PCR	256 (32)	0.0091	0.0045	0.095
Fitting + PLS	30 (29)	0.0112	0.0054	0.106
Fitting + PCR	38 (30)	0.0085	0.0041	0.092
LS-SVM	256	0.0098	0.0047	0.099
Fitting + LS-SVM	32	0.0076	0.0037	0.087
Fitting + LS-SVM + FB	16	0.0058	0.0028	0.076

3.3.3 Juice

Predicting the saccharose content in juice samples is more difficult task than the previous ones, which is clearly seen in Figure 3.7. Looking at the results in Table 3.3, plain LS-SVM does not perform very well, which may be due to the high dimension of the input data. However, after the functional dimensionality reduction, all the three models result in very similar performance, LS-SVM being slightly better. Variable selection improves performance by one fourth, selecting less than half of the variables.

Comparing to the literature, Rossi et. al. have reported a slightly better NMSE 0.081 using LS-SVM with mutual information [24] based variable selection. Benoudjit et. al. obtained NMSE 0.070 with RBFN using FB selection, but similarly to the the Wine data set, the reported NMSE is

Table 3.3: Results of the Juice data set. The number of latent variables are given in parenthesis.

Method	Number of variables	MSE_T	$NMSE_T$	$RMSE_T$
PLS	700 (8)	22.9	0.153	4.78
PCR	700 (14)	22.9	0.153	4.79
Fitting + PLS	27 (27)	18.7	0.125	4.33
Fitting + PCR	27 (27)	19.8	0.132	4.45
LS-SVM	700	32.9	0.220	5.74
Fitting + LS-SVM	31	18.5	0.124	4.30
Fitting + LS-SVM + FB	11	13.9	0.093	3.72

actually validation error.

3.3.4 Discussion

The experimental results on three data sets suggests that dimensionality can be reduced dramatically without loss of prediction accuracy. It was even observed that the functional dimensionality reduction improves results compared to PLS, PCR and plain LS-SVM. Our experiments with the FB selection are very promising in both acquiring better performance and better data compression. Furthermore, the obtained errors are fully comparable to other results reported in literature.

The methodology also seems to be robust since good results were obtained in all cases despite the fairly different nature of the data sets. For example, the LS-SVM model is cabable of dealing with the non-linearities of the Tecator data set. Still, the results on the Juice and the Wine data sets suggests that if the problem is known to be very linear, the Gaussian fitting could be used with standard linear models as well.

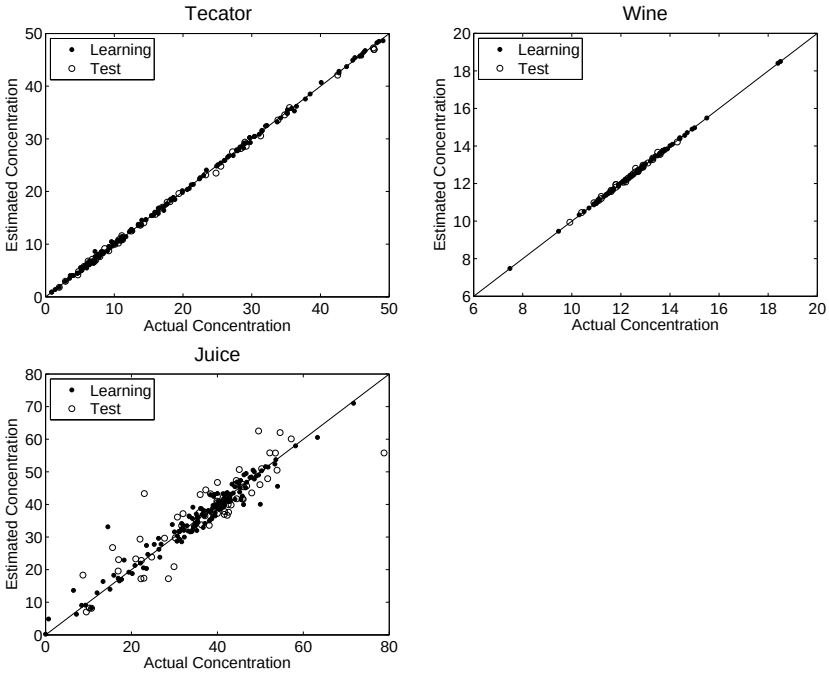

Figure 3.7: Plots of actual concentration versus predicted concentration.

Chapter 4

Application II: Time series prediction

4.1 Introduction

Time series prediction is a classical problem in machine learning. In recent years it has received considerable attention in many fields such as climatology, finance and energy supply management. In this work, we will restrict our discussion to reguarly sampled time series and to prediction models that have scalar output, i.e. are able to predict only one value at a time. Generally speaking, in this case, there are two main strategies for long-term forecasts. The first approach is to build a one step ahead prediction model which is then applied recursively to obtain longer predictions. The drawback of this recursive prediction is that errors tend to accumulate as already predicted values are used as new inputs. Another approach, that is often referred to as direct prediction, is to build separate models for each unknown value. Therefore the accumulation of errors is avoided but computational costs may become an issue as the prediction horizon grows: a L step ahead prediction requires L independent models to be built. In most cases the direct prediction method yields better results, which is generally due to its higher complexity [36].

This chapter presents a modification to the direct prediction schema. The Gaussian fitting is used to reduce the dimensionality of the output, i.e. the L target values are expressed in a more compact form and thus the number of required prediction models is reduced.

The methodology is experimented with two important applications of time

series prediction. The first application is the ESTSP'07 data set, which concerns prediction of Sea Surface Temperature (SST) in Pacific Ocean and is thus an example of climatological forecasting. The development of SST is an essential factor in the interannual warming and cooling anomaly of ocean waters in the Pacific known as the El Niño–Southern Oscillation (ENSO). ENSO has a major impact on the climate in global scale which is why there has been substantial interest in forecasting the SST anomalies [37, 38, 39, 40].

The second application is electricity load prediction, which has become an important tool for energy supply operators. Accurate load forecasts can lead to significant economical savings because the output of power plants can be regulated to meet the actual demand. The most commonly used prediction methods are Artificial Neural Networks (ANN) and Expert Systems (ES) [41].

For both of the applications, the outputs are compressed by the functional dimensionality reduction that was introduced in Section 2.3. Again LS-SVM regression model is used for the prediction. For comparison, results obtained with k-NN model are presented as well.

Similarly to the chemometrics application, compressing the input data is also experimented. This is motivated by the fact that in long-term time series prediction the input data dimension tends to be large which may cause problems due to the curse of dimensionality [2, 3].

The chapter is organised as follows. Traditional time series prediction methodology is introduced in Section 4.2 and its modification that utilises functional inputs and outputs is presented in Section 4.3. Section 4.4 presents the two applications with results. Finally, some summarising notions are given in Section 4.4.3.

4.2 Conventional time series prediction

Consider a time series $\{y_i\}_{i=1}^N$ measured with a constant sampling rate. In long-term prediction the goal is to build a model for L_O step ahead prediction. The simplest approach is recursive precition where a one step ahead model is used recursively:

$$\begin{aligned}
\hat{y}_{t+1} &= f(y_t, y_{t-1}, \ldots, y_{t-L_I+1}) \\
\hat{y}_{t+2} &= f(\hat{y}_{t+1}, y_t, \ldots, y_{t-L_I+2}) \\
&\vdots \\
\hat{y}_{t+L_O} &= f(\hat{y}_{t+L_O-1}, \hat{y}_{t+L_O-2}, \ldots, y_{t-L_I+L_O})
\end{aligned}$$

The model f operates with input window of L_I previous values, including the previously obtained predictions. Thus the errors are accumulating as the prediction horizon grows. An alternative approach is direct prediction where predicted values are not used as inputs. In this case, the input window is kept fixed and separate models are built for all the outputs:

$$\begin{aligned} \hat{y}_{t+1} &= f_1(y_t, y_{t-1}, \ldots, y_{t-L_I+1}) \\ \hat{y}_{t+2} &= f_2(y_t, y_{t-1}, \ldots, y_{t-L_I+1}) \\ &\vdots \\ \hat{y}_{t+L_O} &= f_{L_O}(y_t, y_{t-1}, \ldots, y_{t-L_I+1}). \end{aligned}$$

Naturally, the accumulation of errors is avoided but the drawback is that L_O separate models need to be trained. Generally, it can be said that the direct prediction method performs better, which is due to the fact that it is L_O-times more complex than the equivalent recursive model.

When working with the direct prediction strategy in practice, the time series is divided into input windows I_i and strictly following output windows O_i:

$$\begin{aligned} I_i &= \{(x_h, z_i^h) \mid \ x_h = h, \ z_i^h = y_{h+(i-1)d}, \ h = 1, \ldots, L_I\} \\ O_i &= \{(x_h, z_i^h) \mid \ x_h = h, \ z_i^h = y_{L_I+h+(i-1)d}, \ h = 1, \ldots, L_O\} \\ i &= 1, \ldots, n, \quad n = \lfloor (N - L_O - L_I)/d \rfloor + 1 \end{aligned}$$

Notice that since the time series is regularly sampled, the arguments (or time stamps) x_h do not depend on the window. As mentioned in Section 2.3, the x_h can be chosen freely and $x_h = h$ is a natural choice.

Using this notation, the prediction problem is equivalent to estimating the O_i values based on the corresponding I_i. The delay parameter d determines how much time shift there is between different windows. Often d is set to one in order to maximise the number of training samples. However, in the case of periodic time series, it can be set to the cycle length to ensure that all the windows share the same periodic structure.

This chapter presents a modification to the direct prediction scheme where the output windows O_i are expressed in more compact form in order to reduce the number of required prediction models. The compact representation is obtained using the functional dimension reduction.

4.3 Time series prediction in function space

The functional dimensionality reduction can be applied to inputs windows I_i or output windows O_i. To introduce the notation, it is assumed that it is applied to both of them. It should be noted that because the input and output windows differ in length, one needs to build separate basis functions for the two cases.

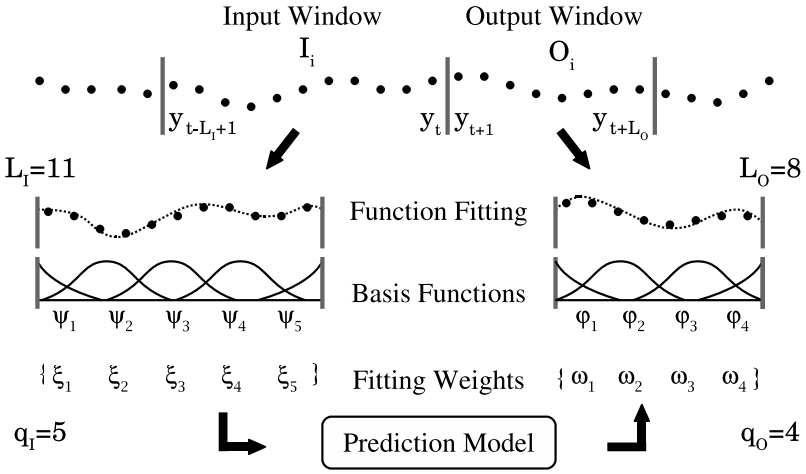

Figure 4.1: Time series prediction in the function space. The prediction model operates with the fitting coefficients instead of the original data points. The number of prediction models that need to be built equals to the dimension of the output space, q_O.

A flow-graph of the prediction scheme is presented in Figure 4.1. Denote the basis of input widows by $\psi_j(x),\ j = 1, \ldots, q_I$ and the basis of output windows by $\varphi_j(x),\ j = 1, \ldots, q_O$. As before, the bases consist of Gaussian functions that are optimised to fit the data, as explained in Sections 2.3.1 and 2.3.2. Now, time series data can be expressed in functions, namely $u_i(x) = \boldsymbol{\xi}_i^T \boldsymbol{\psi}(x)$ and $v_i(x) = \boldsymbol{\omega}_i^T \boldsymbol{\varphi}(x)$ for I_i and O_i, respectively. Here we use the subscript i to emphasise that the fitting weights depend on the window, while the basis functions are kept fixed for all i.

Using the functional representations (i.e. the weights $\boldsymbol{\xi}_i = [\xi_i^1, \xi_i^2, \ldots, \xi_i^{q_I}]^T$ and $\boldsymbol{\omega}_i = [\omega_i^1, \omega_i^2, \ldots, \omega_i^{q_O}]^T$), the direct prediction strategy can be reformulated as

$$\hat{\omega}_i^j = f_j(\xi_i^1, \xi_i^2, \ldots, \xi_i^{q_I}),\ j = 1, \ldots, q_O. \tag{4.1}$$

These prediction models operate solely in the function space. It should be

stressed that the data dimension of both of the windows have been reduced, i.e. $q_I < L_I$ and $q_O < L_O$, which implies that not only the number of prediction models is reduced but the models also operate in lower dimensional space.

Since the prediction is carried out in the function space, the goal is to estimate the fitting weights $\hat{\boldsymbol{\omega}}_i$ as accurately as possible. In other words, for each model f_j, the objective is to minimise the error $E(f_j) = 1/n \sum_{i=1}^{n} (\hat{\omega}_i^j - \omega_i^j)^2$. Naturally this is not equivalent to minimising the prediction error in time space. Nevertheless, one can argue that when the error in function space tends to zero, the error in the time space approaches the smallest fitting error. Furthermore, training the models in the function space implies that standard training algorithms can be used and is there is no need to define a special cost function that might be computationally expensive.

After the training is complete, one needs to determine how well the trained models actually perform in the time space. For this purpose, the estimated functions $\hat{v}_i(x) = \hat{\boldsymbol{\omega}}_i^T \boldsymbol{\varphi}(x)$ are evaluated at the data locations and compared to the target output values O_i. The final prediction errors are expressed in NMSE, which is now defined as:

$$\text{NMSE} = \frac{1}{\text{Var}(y_j)} \frac{1}{nL_O} \sum_{i=1}^{n} \sum_{(x_h, z_i^h) \in O_i} (z_i^h + \hat{\boldsymbol{\omega}}_i^T \boldsymbol{\varphi}(x_h))^2, \tag{4.2}$$

where $\text{Var}(y_j)$ is the variance of the original time series.

The proposed methology involves some parameters that need to be determined when working with a practical application. The desired prediction horizon (i.e. L_O) is usually known a priori. Selection of the output space dimension, q_O, is two fold: High dimensional function fitting usually provides better performance, but at the expense of computational time. However, the value of q_O is not very crucial for the overall performance and therefore it is suggested that it can be chosen heuristically: It is enough to set q_O sufficiently large to quarantee a "good enough" fitting error, such as a certain fraction of the desired prediction error. The input window parameters, however, play an important role in the quality of the prediction and are more difficult to infer a priori. Therefore L_I and q_I should be validated during the training process.

This section has concentrated on an approach where the functional dimensionality reduction is applied to both inputs and outputs. In practice, however, the validation of the input window parameters can be time consuming and as an alternative one can use the raw time series inputs instead. Thus there is no need to build the basis nor select the parameters for the input

windows, but still major savings in computational time can achieved due to the smaller output dimension.

4.4 Experiments

4.4.1 ESTSP'07 benchmark data

The first application concerns the ESTSP'07 contest data. The data set consists of weekly Sea Surface Temperatures (SST) measured in the Pacific Ocean near coast of Peru during years 1990-2007. Traditionally the STT indexes have been forecasted by building physical models of the ocean-atmosphere system (such as the one presented in [37]), but also by statistical models [38, 39] and Artificial Neural Networks (ANN) [40]. Since the SST is determined by a complex ocean-atmosphere interaction, the non-physical models are sometimes coupled with other climatological data, such as wind stress [40] or sea level air pressure [39]. In this work, however, only time series information is utilised.

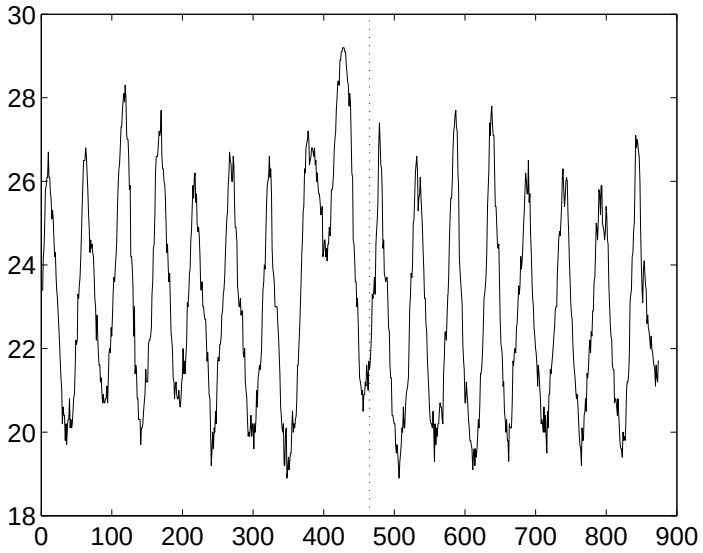

Figure 4.2: ESTSP'07 data set, monthly average SST measured in the Pacific Ocean between years 1990-2007.

The data is illustrated in Figure 4.2. Notice that the 1997/1998 El Niño

(warm anomaly) is clearly visible. First 465 values were used as a learning set and the latter 410 as a test set. The goal in the ESTSP'07 competition was to build models for 15 steps and 50 steps ahead prediction.

The data was sliced into windows as explained in Section 4.2. The delay between windows was 1, because there is no structure in the data that could be exploited.

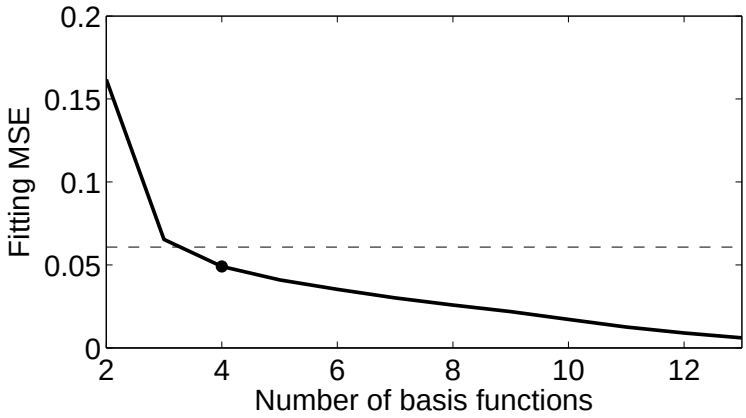

Figure 4.3: ESTSP: Fitting accuracy of output windows in 15 steps ahead prediction. The dashed line stands for 0.01 NMSE threshold level.

The basis functions were optimised as explained in Section 2.3.2. In the case of the output windows, the number of basis functions was selected by the quality of the fitting. The fitting error versus number of basis functions is plotted in Figures 4.3 and 4.4 for 15 steps and 50 steps ahead prediction, respectively. The threshold level was set to 0.01 NMSE, which was considered as sufficient accuracy. This yielded 4 basis functions for 15 steps ahead and 10 for 50 steps ahead prediction. Thus, in both cases the function fitting yielded significant dimensionality reduction. The basis functions and examples of approximated functions are plotted in Figures A.4 and A.5.

In the case of input windows, the window length and the number of basis functions were selected using LS-SVM validation error. The fitting accuracy versus number of basis functions are presented in Figures 4.5 and 4.6.

LS-SVM model was trained in the function space using two dimensional grid search and 10-fold cross-validation. Furthermore, LS-SVM prediction was experimented with a setting where the function approximation was used only at the output window. In this case the inputs were the plain time series data points in I_i.

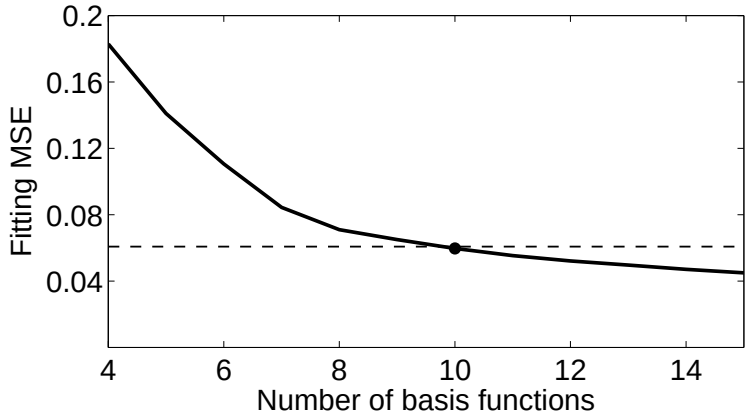

Figure 4.4: ESTSP: Fitting accuracy of output windows in 50 steps ahead prediction. The dashed line stands for 0.01 NMSE threshold level.

Since a single k-NN model is able produce multidimensional output, we did not use the function fitting in the output windows in this case, because it would not introduce any clear benefits. The input window length and the number of basis functions were selected using the k-NN LOO validation error.

Finally, the results were compared to the conventional direct prediction (i.e. prediction without any function fitting) that was carried out using both k-NN and LS-SVM.

Results

The results of 15 steps and 50 steps ahead prediction are presented in Tables 4.1 and 4.2, respectively.

Table 4.1: Results of the ESTSP 15 steps ahead prediction.

Model	Input	Output	MSE_T	$NMSE_T$
LS-SVM	Function	Function	1.03	0.17
LS-SVM	Plain	Function	0.98	0.16
LS-SVM	Plain	Plain	**0.98**	**0.16**
k-NN	Function	Plain	1.44	0.24
k-NN	Plain	Plain	1.58	0.26

In 15 steps ahead prediction, the prediction errors of the LS-SVMs are quite

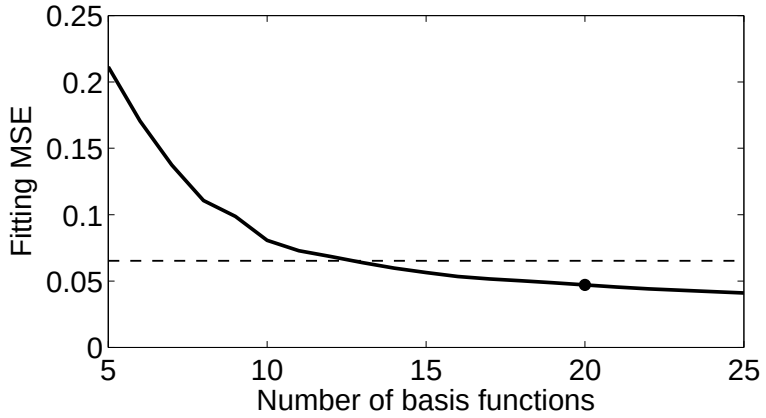

Figure 4.5: ESTSP: Fitting accuracy of input windows in 15 steps ahead prediction. The input window length is 74. Both the input window length and number of basis functions were validated using LS-SVM 10-fold cross validation.

similar. Plain LS-SVM performs the best, but the differences are very small. In the case of k-NN, functional inputs give slightly better results than plain inputs although both are clearly worse than any of the LS-SVMs.

The differences are clearer in 50 steps ahead prediction. In this case, the k-NN models are already rather poor. The completely functional LS-SVM results in NMSE 0.22 corresponding to $R^2 = 0.78$, which is quite satisfactory. The plain LS-SVM a bit worse.

Table 4.2: Results of the ESTSP 50 steps ahead prediction.

Model	Input	Output	MSE_T	$NMSE_T$
LS-SVM	Function	Function	**1.33**	**0.22**
LS-SVM	Plain	Function	2.30	0.38
LS-SVM	Plain	Plain	1.74	0.29
k-NN	Function	Plain	3.73	0.61
k-NN	Plain	Plain	3.21	0.53

Examples of predicted curves can be seen in Figures 4.7 and 4.8 for 15 steps and 50 steps ahead prediction, respectively. The presented models are plain k-NN, plain LS-SVM and LS-SVM with functional inputs and outputs. Note that due to the dimension reduction the functional LS-SVM produces

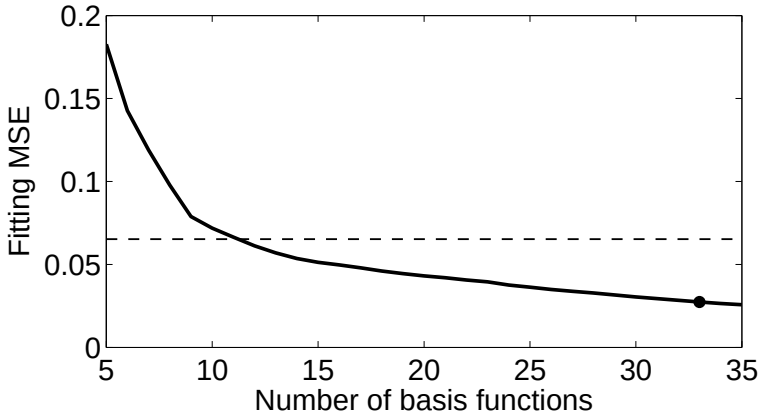

Figure 4.6: ESTSP: Fitting accuracy of input windows in 50 steps ahead prediction. The input window length is 65. Both the input window length and number of basis functions were validated using LS-SVM 10-fold cross validation.

smoother curves than the other methods. Plain k-NN seems to be most noisy prediction method.

All in all, it can be said that the ESTSP competition data is not very easy to predict. This is most likely due to two reasons: First, the SST variation is a consequence of complex ocean-atmosphere interaction and therefore locally measured time series itself does not contain all the necessary information. Second, as usual with climatological data, the cycle length is long compared to the amount of data. The entire data set consist of 17 years of measurements while the ENSO anomaly usually has period from 3 to 5 years [38].

4.4.2 Electricity consumption prediction

The electricity data set contains consumption of mainland France measured between years 1996-2005 with 30 minutes temporal resolution summing up to a total 175344 values. The data are plotted in Figure 4.9 where the rising trend in total consumption is clearly visible. Naturally, the data are periodic and due to the high temporal resolution one can distinguish three different cycles: daily, weekly and yearly. The data can be freely downloaded from the RTE France website: `http://www.rte-france.com/`.

Electricity consumption is highly dependent on external factors, such as

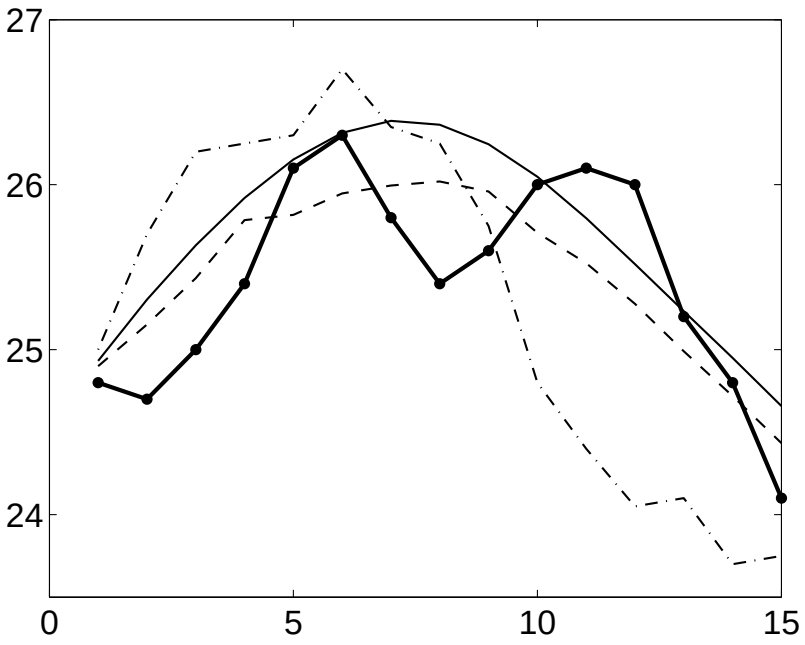

Figure 4.7: ESTSP: Example of 15 steps ahead prediction. Original data is marked with bold line, thin solid line stands for the functional LS-SVM, dashed line for plain LS-SVM and dash-dotted line for plain k-NN.

weather conditions and national holidays. Therefore, exogenous information is often coupled with the actual time series data in order to improve the prediction results. Yalcinoz et. al [42] used daily average temperatures with a Multi-Layer Preceptron (MLP) model for daily load forecasting. Beccali et. al [43] have presented another MLP based method that incorporates also hourly temperature, solar radiation and relative humidity measurements. However, in this study, we consider the problem as a pure time series prediction and do not use any external information.

As before, the data was sliced into windows, but in this case the delay was set to one day (i.e. 48 values) and the window lengths were also restricted to whole days. This implies that all the windows share similar daily periodicity. The choice was motivated by the fact that the functional dimension reduction becomes more efficient when all the windows have similar curves. The downside is that the total number of windows is reduced but it is not a major problem since there are plenty of data.

First, a simple one day ahead prediction was experimented. Again the threshold for the output window fitting quality was set to 0.01 NMSE, which re-

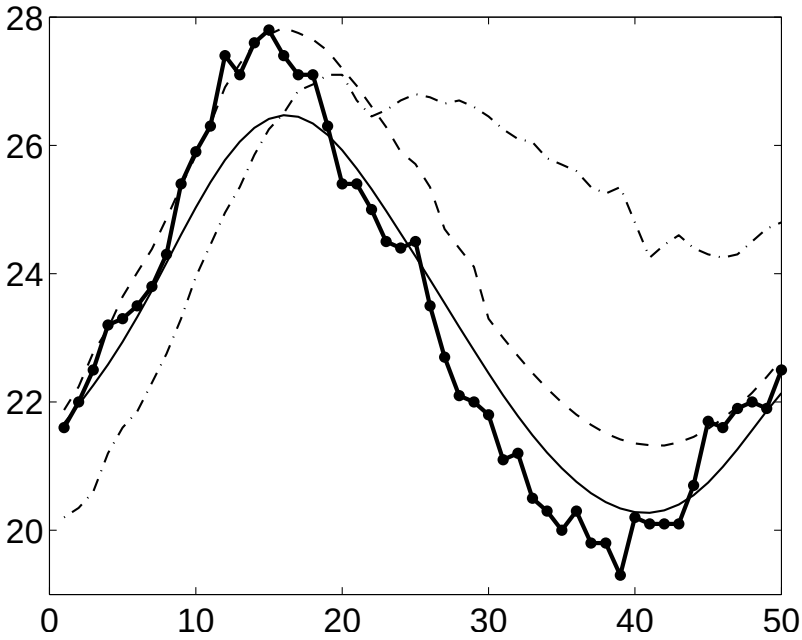

Figure 4.8: ESTSP: Example of 50 steps ahead prediction. Original data is marked with bold line, thin solid line stands for the functional LS-SVM, dashed line for plain LS-SVM and dash-dotted line for plain k-NN.

sulted in seven basis functions (see Figure 4.10). The seven basis functions and an example of fitted function is presented in Figure A.6. The input window length ranged from 48 (one day) to 336 (one week) out of which 240 was selected. Figure 4.11 illustrates the fitting error versus number of basis functions. Also in this case we experimented direct prediction, prediction with functional outputs and prediction with both functional inputs and outputs. Note that the direct LS-SVM prediction involves 48 separate models, which is already quite time consuming.

To test whether the methodology is suitable for to very long prediction horizons, we also experimented one week, i.e. 336 steps ahead prediction. Clearly, in this case the conventional direct prediction strategy is already out of question: Training 336 separate LS-SVM models would take too long even with the most up-to-date computers. Reaching the desired 0.01 NMSE fitting quality in the output windows would have required 41 basis functions (see Figure 4.12). In order to decrease the computational complexity, 25 basis functions and fitting quality of NMSE 0.03 was selected. The basis functions are illustrated in Figure A.7. Input window length ranged from one week to

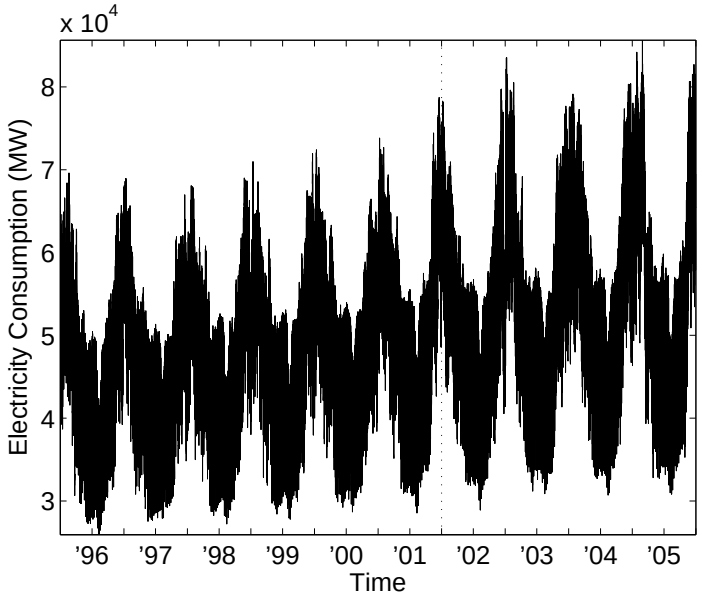

Figure 4.9: RTE electricity consumption data. Consumption of mainland France in *MW*.

two weeks. The input window fitting accuracy is presented in Figure 4.13.

Results

Results of the electricity consumption prediction are presented in Tables 4.3 and 4.4 for one day and one week ahead prediction, respectively.

Table 4.3: Results of the RTE 48 steps ahead prediction.

Model	Input	Output	MSE_T	$NMSE_T$
LS-SVM	Function	Function	3.9	0.040
LS-SVM	Plain	Function	3.8	0.039
LS-SVM	Plain	Plain	**3.0**	**0.031**
k-NN	Function	Plain	14.3	0.149
k-NN	Plain	Plain	11.0	0.114

In one day ahead prediction, all the LS-SVMs perform very well resulting in NMSE 0.04 or less. On the other hand, k-NN is clearly worse: the NMSE is

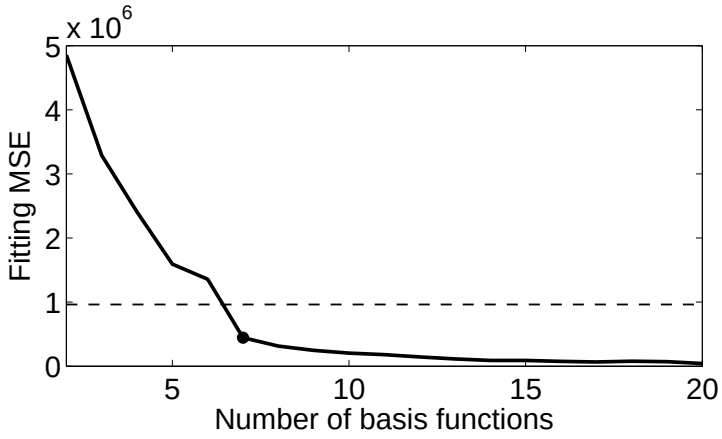

Figure 4.10: RTE: Fitting accuracy of output windows in 48 steps ahead prediction. The dashed line stands for 0.01 NMSE threshold level.

roughly 3-fold.

Although direct LS-SVM is the best model, it is presented here mainly for comparison rather than a suggestion for a practical implementation. The RTE data set is very large and thus the direct prediction scheme is computationally demanding: Computing the entire validation process, including the 48 LS-SVM models, took roughly 8 days on a modern single-CPU computer. In contrast, the computational time for plain-input functional-output LS-SVM was less than 12 hours on the same machine.

The performance of LS-SVM with functional outputs is also very good. Interestingly enough, coupling functional inputs together with functional outputs seems to make no significant difference. Therefore, one should favour the simpler model, i.e. the plain inputs.

Table 4.4: Results of the RTE 336 steps ahead prediction.

Model	Input	Output	MSE_T	$NMSE_T$
LS-SVM	Function	Function	**13.4**	**0.14**
LS-SVM	Plain	Function	14.3	0.15
LS-SVM	Plain	Plain	– –	– –
k-NN	Function	Plain	23.5	0.24
k-NN	Plain	Plain	19.4	0.20

The results of one week ahead prediction are well in line with the previous

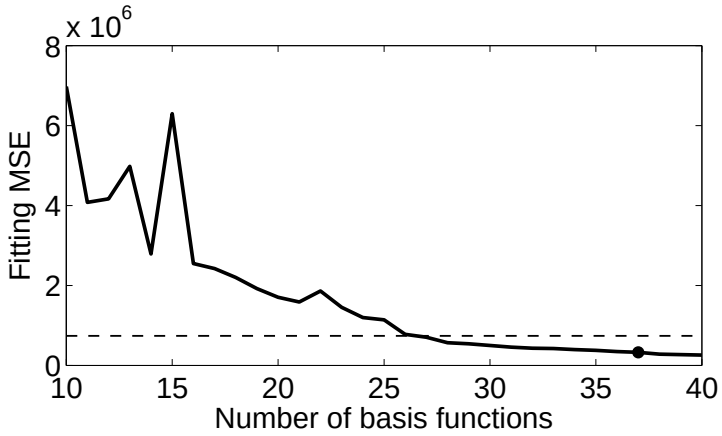

Figure 4.11: RTE: Fitting accuracy of input windows in 48 steps ahead prediction. The input window length is 240. Both the input window length and number of basis functions were validated using LS-SVM 10-fold cross validation.

notions. The two direct LS-SVMs with functional outputs are again almost equally good. The k-NN models are worse but the difference is not as grave as before. The best models yield NMSE of about 0.14 which is entirely satisfactory considering the length of the prediction.

Examples of one day and one week ahead predictions are presented in Figures 4.14 and 4.15. The smoothing effect of the function fitting is even more apparent: Functional LS-SVM does not predict the narrow spikes of the data, rather it produces a general trend of the future behaviour.

Comparing to the ESTSP'07 data set, the RTE data set is very large which is probably one reason for the good results. It was also observed that, especially in the one week ahead prediction, there are plenty of examples that all the models are able to predict with good accuracy. The differences in performace are therefore due to more difficult cases, such as the one presented in Figure 4.15. This kind of unexpected variation is most likely due to changes in the external conditions which suggests that the models could be improved by utilising external information.

4.4.3 Discussion

In this work, a functional dimensionality reduction method for speeding up the direct time series prediction methodology has been experimented. The

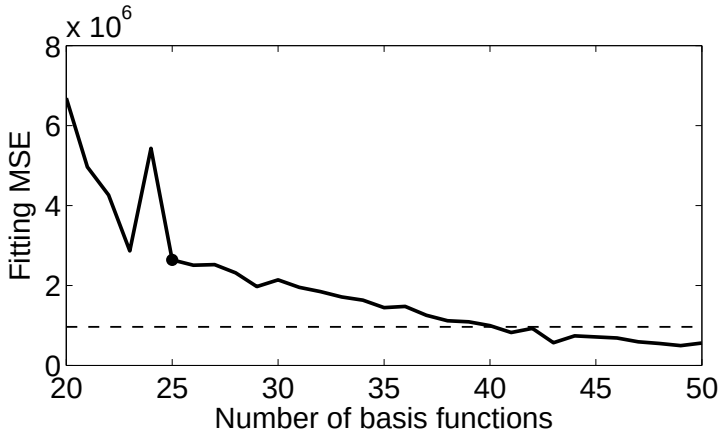

Figure 4.12: RTE: Fitting accuracy of output windows in 336 steps ahead prediction. The dashed line stands for 0.01 NMSE threshold level. However, in order to reduce computational time, 25 functions were chosen.

results obtained with the two data sets show that the proposed methodology indeed performs very well and the advantages become clearer as the length of the prediction grows. Indeed, predicting electricity consumption with LS-SVM 336 steps ahead would have been impossible without the functional dimensionality reduction. Already in the case of 48 steps ahead prediction the plain LS-SVM model required 16 times more computation time than the reduced model. And what is most important, the obtained results were satisfactory and clearly better compared to k-NN. Therefore LS-SVM seems to be more suitable for long-term prediction than k-NN despite the fact that k-NN is able to deal with multidimensional outputs directly.

Nonetheless, based on these results, it is not clear whether the functional data compression should be used for inputs as well. It was clearly beneficial only in the case of ESTSP 50 steps ahead prediction. This suggests that in many cases it is enough to compress only the outputs and thus simplify the methodology.

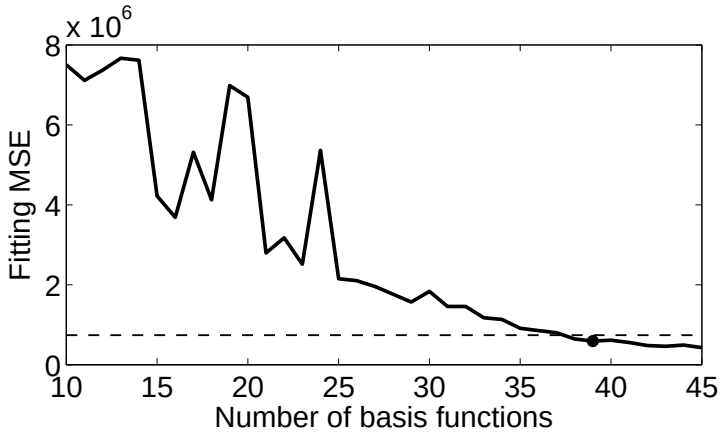

Figure 4.13: RTE: Fitting accuracy of input windows in 336 steps ahead prediction. The input window length is 336. Both the input window length and number of basis functions were validated using LS-SVM 10-fold cross validation.

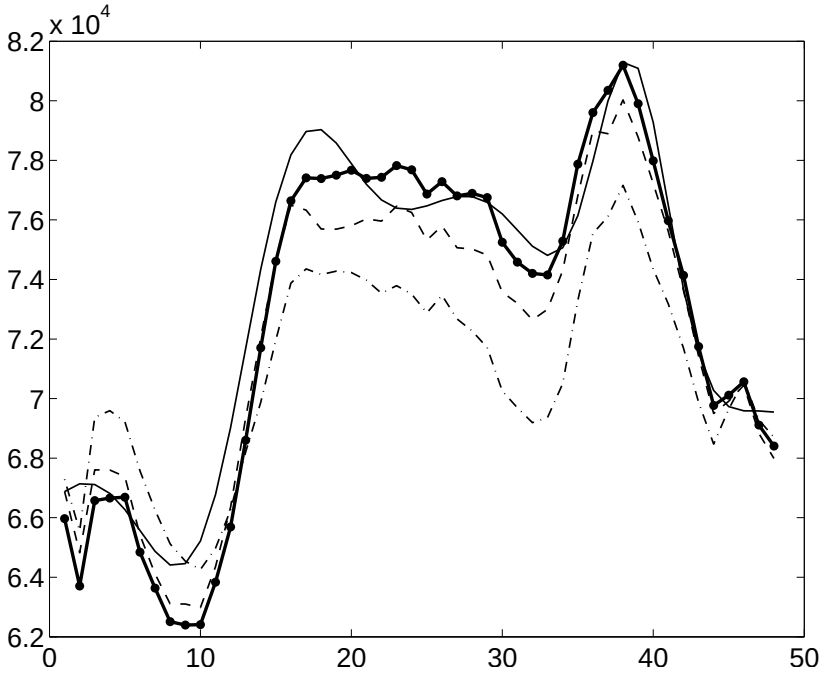

Figure 4.14: RTE: Example of one day (48 steps) ahead prediction. Original data is marked with bold line, thin solid line stands for the functional LS-SVM, dashed line for plain LS-SVM and dash-dotted line for plain k-NN.

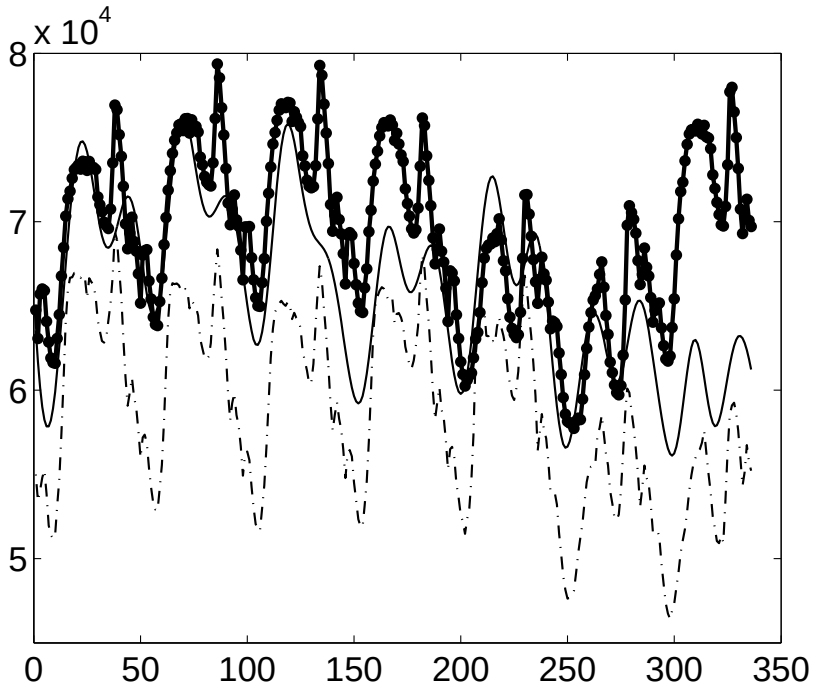

Figure 4.15: RTE: Example of one week (336 steps) ahead prediction. Original data is marked with bold line, thin solid line stands for the functional LS-SVM and dash-dotted line for plain k-NN.

Chapter 5

Conclusions

Dimensionality reduction is an important area of data analysis because it assists in circumventing the curse of dimensionality, reduces computational time and aids in interpreting the data. This thesis concentrates on functional data dimensionality reduction for regression problems which are often encountered in the field of machine learning.

The dimensionality reduction is obtained by projecting the data onto a lower dimensional function space that is defined by a set of basis functions. The advantage of the proposed methodology is that the basis can be adjusted to suit a specific problem through non-linear optimisation process. Thus an accurate representation of the data is obtained with a few basis functions leading into efficient dimensionality reduction.

As mentioned in the Introduction, the functional dimensionality reduction is more restricted compared to the general non-linear projection methods because the functional representation is not applicable to all kinds of data. Still, the experimental results show that the methodology can be successfully applied to cases where the data is smooth enough to be represented as a function. The obtained data compression ratios are given in Table 5.1. In the case of chemometrics, the dimension of the function space was validated during the training process. In time series prediction, on the other hand, the dimensionality of the outputs was selected based on a heuristic error threshold.

The experimental results obtained with spectral data are good both in terms of data compression and prediction performance. The reason behind this is that spectra are well suited for functional dimensionality reduction: First of all, the spectral curves are typically very smooth. In addition, in chemometrics the spectra of different samples are very similar (obviously due to

Table 5.1: Data compression ratios in the tested cases.

Data set	Original dimension	Compression ratio	Compression ratio after FB selection
ESTSP	15	3.75	–
ESTSP	50	5.00	–
RTE	48	6.86	–
RTE	336	13.44	–
Tecator	100	6.67	9.09
Wine	256	8.00	16.00
Juice	700	22.58	63.64

the fact that the samples are similar) which implies that the intrinsic data dimension is small despite the high accuracy of the spectrometer.

One clear advantage of the smaller data dimension is the reduced computational time. Indeed, in chemometrics variable selection is a common yet computationally demanding method which can be significantly simplified by the functional dimensionality reduction. Although the selection is carried out in the function space and does not therefore involve the wavelengths directly, it can still provide useful information about the regression task: The basis consists of local functions and therefore the selected functions indicate which parts of the spectrum are important for the prediction.

Compared to chemometrics, similar data compression ratios are more difficult to achieve in other applications where the data have no clear structure or are corrupted by noise. In time series prediction, both of these properties often hold. Nevertheless, compressing the output dimension of the model offer clear advantages in speeding up the direct prediction methodology. In fact, as the electricity consumption application demonstrated, it is impossible to utilise the direct prediction schema in very long prediction horizons without reducing the dimensionality. Furthermore, if the time series is periodic, the local windows can be chosen so that they share common periodic structure. Thus the basis adapts to the cyclic waveform which increases the efficiency of dimension reduction. Finally it should be noted that although this thesis has covered only regularly sampled time series, the methodology could be applied to irregular time series as well with minor modifications [11].

In both of the applications it was observed that the dimensionality reduction is more efficient with high dimensional data. Therefore, one can conclude

that the proposed functional dimensionality reduction method is especially suitable for high dimensional data that are relatively smooth and have some intrinsic structure.

Bibliography

[1] S. Haykin. *Neural Networks: A Comprehensive Foundation.* Prentice Hall, 2nd edition, 1999.

[2] M. Verleysen and D. François. The curse of dimensionality in data mining and time series prediction. In J. Cabestany, A. Prieto, and D.F. Sandoval, editors, *8th International Work-Conference on Artificial Neural Networks*, volume 3512 of *Lecture Notes in Computer Science*, pages 758–770. Springer-Verlag, 2005.

[3] K. Beyer, J. Goldstein, R. Ramakrishnan, and U. Shaft. When is "nearest neighbor" meaningful? In C. Beeriand and P. Buneman, editors, *The 7th International Conference on Database Theory*, volume 1540 of *Lecture Notes in Computer Science*, pages 217–235. Springer-Verlag, 1998.

[4] C. Bishop. *Neural Networks for Pattern Recognition.* Oxford University Press, 1995.

[5] R. Rosipal and L.J. Trejo. Kernel partial least squares regression in reproducing kernel hilbert space. *Journal of Machine Learning Research*, 2:97–123, 2001.

[6] Y. Bengio, O. Delalleau, and N. Le Roux. The curse of highly variable functions for local kernel machines. In Y. Weiss, B. Schölkopf, and J. Platt, editors, *Neural Information Processing Systems*, volume 18, pages 107–114, 2006.

[7] P. Demartines and J. Herault. Curvilinear component analysis: a self-organizing neural networkfor nonlinear mapping of data sets. *IEEE Transactions on Neural Networks*, 1997.

[8] J.B. Tenenbaum, V. de Silva, and J.C. Langford. A global geometric framework for nonlinear dimensionality reduction. *Science*, 2000.

[9] M. Belkin and P. Niyogi. Laplacian eigenmaps for dimensionality reduction and data representation. *Neural Computation*, 2003.

[10] J. Ramsay and B. Silverman. *Functional Data Analysis*. Series in statistics. Springer Verlag, 1997.

[11] T. Kärnä, F. Rossi, and A. Lendasse. LS-SVM functional network for time series prediction. In M. Verleysen, editor, *14th European Symposium On Artificial Neural Networks*, pages 473–478, 2006.

[12] T. Kärnä and A. Lendasse. Comparison of FDA based time series prediction methods. In *European Symposium on Time Series Prediction*, pages 77–86, 2007.

[13] T. Kärnä and A. Lendasse. Gaussian fitting based FDA for chemometrics. In F. Sandoval, A. Prieto, J. Cabestany, and M. Gra na, editors, *9th International Work-Conference on Artificial Neural Networks*, volume 4507 of *Lecture Notes in Computer Science*, pages 186–193. Springer-Verlag, 2007.

[14] T. Kärnä, F. Corona, and A. Lendasse. Gaussian basis functions for chemometrics. Submitted to *Journal of Chemometrics*.

[15] T. Kärnä and A. Lendasse. Functional dimensionality reduction for long-term time series prediction. Submitted to *Neurocomputing*.

[16] J. Kaipio and E. Somersalo. *Statistical and Computational Inverse Problems*. Springer Verlag, 2004.

[17] B. Efron and R.J. Tibshirani. *An Introduction to the Bootstrap*. Chapman and Hall, 1994.

[18] J. Suykens, T. Van Gestel, J. De Brabanter, B. De Moor, and J. Vandewalle. *Least Squares Support Vector Machines*. World Scientific Publishing, 2002.

[19] A. Sorjamaa, N. Reyhani, and A. Lendasse. Input and structure selection for k-NN approximator. In J. Cabestany, A. Prieto, and D.F. Sandoval, editors, *8th International Work-Conference on Artificial Neural Networks*, volume 3512 of *Lecture Notes in Computer Science*, pages 985–991. Springer-Verlag, 2005.

[20] F. Rossi, N. Delannay, B. Conan-Guez, and M. Verleysen. Representation of functional data in neural networks. *Neurocomputing*, 64:183–210, 2005.

[21] C. de Boor. *A Practical Guide to Splines*, volume 27 of *Applied Mathematical Sciences*. Springer-Verlag, 1978.

[22] I. Daubechies. *Ten Lectures on Wavelets*, volume 61 of *CBMS-NSF Regional Conference Series in Applied Mathematics*. Society for Industrial and Applied Mathematics, 1992.

[23] M.S. Bazaraa, H.D. Sherali, and C.M. Shetty. *Nonlinear Programming, Theory and Algorithms*. John Wiley and Sons, 2nd edition, 1993.

[24] F. Rossi, A. Lendasse, D. François, V. Wertz, and M. Verleysen. Mutual information for the selection of relevant variables in spectrometric nonlinear modelling. *Chemometrics and Intelligent Laboratory Systems*, 80:215–226, 2006.

[25] W. Härdle, H. Liang, and J.T. Gao. *Partially Linear Models*. Physica-Verlag, 2000.

[26] E. Vigneau, M.F. Devaux, E.M. Qannari, and P. Robert. Principal component regression, ridge regression and ridge principal component regression in spectroscopy calibration. *Journal of Chemometrics*, 11:239–249, 1997.

[27] F. Wülfert, W.T. Kok, and A.K. Smilde. Influence of temperature on vibrational spectra and consequences for the predictive ability of multivariate models. *Analytical Chemistry*, 70:1761–1767, 1998.

[28] F. Chauchard, R. Cogdill, S. Roussel, J.M. Roger, and V. Bellon-Maurel. Application of LS-SVM to non-linear phenomena in NIR spectroscopy: development of a robust and portable sensor for acidity prediction in grapes. *Chemometrics and Intelligent Laboratory Systems*, 71:141–150, 2004.

[29] B.K. Alsberg and O.M. Kvalheim. Compression of nth-order data arrays by b-splines. I : Theory. *Journal of Chemometrics*, 7:61–73, 1993.

[30] X.G. Shao, A.K. Leung, and F.T. Chau. Wavelet: A new trend in chemistry. *Accounts of Chemical Research*, 36:276–283, 2003.

[31] J. Trygg and S. Wold. PLS regression on wavelet compressed NIR spectra. *Chemometrics and Intelligent Laboratory Systems*, 42:209–220, 1998.

[32] C. Borggaard and H. Thodberg. Optimal minimal neural interpretation of spectra. *Analytical Chemistry*, 64:545–551, 1992.

[33] H. Thodberg. A review of bayesian neural networks with an application to near infrared spectroscopy. *IEEE Transactions on Neural Networks*, 7:56–72, 1996.

[34] N. Benoudjit, E. Cools, M. Meurens, and M. Verleysen. Chemometric calibration of infrared spectrometers: selection and validation of variables by non-linear models. *Chemometrics and Intelligent Laboratory Systems*, 70:47–53, 2004.

[35] J. Vila, V. Wagner, and P. Neveu. Bayesian nonlinear model selection and neural networks: A conjugate prior approach. *IEEE Transactions on Neural Networks*, 11:265–278, 2000.

[36] A. Sorjamaa, J. Hao, N. Reyhani, Y. Ji, and A. Lendasse. Methodology for long-term prediction of time series. *Neurocomputing*, 70(16-18):2861–2869, 2007.

[37] T.P. Barnett, N. Graham, S. Pazan, W. White, M. Latif, and M. Flügel. ENSO and ENSO-related predictability. part I: Prediction of equatorial pacific sea surface temperature with a hybrid coupled ocean–atmosphere model. *Journal of Climate*, 6(8):1545–1566, 1993.

[38] L.M. Berliner, C.K. Wikle, and N. Cressie. Long-lead prediction of pacific SSTs via bayesian dynamic modeling. *Journal of Climate*, 13(22):3953–3968, 2000.

[39] Y. Xue, A. Leetmaa, and M. Ji. ENSO prediction with markov models: The impact of sea level. *Climate Dynamics*, 9(4-5):167–179, 1994.

[40] F. T. Tangang, W. W. Hsieh, and B. Tang. Forecasting the equatorial pacific sea surface temperatures by neural network models. *Climate Dynamics*, 13:135–147, 1997.

[41] K. Metaxiotis, A. Kagiannas, D. Askounis, and J. Psarras. Artificial intelligence in short term electric load forecasting: a state-of-the-art survey for the researcher. *Energy Conversion and Management*, 44:1525–1534, 2003.

[42] T. Yalcinoz and U. Eminoglu. Short term and medium term power distribution load forecasting by neural networks. *Energy Conversion and Management*, 46:1393–1405, 2005.

[43] M. Beccali, M. Cellura, V. Lo Brano, and A. Marvuglia. Forecasting daily urban electric load profiles using artificial neural networks. *Energy Conversion and Management*, 45:2879–2900, 2004.

Appendix A

Examples of optimised basis functions

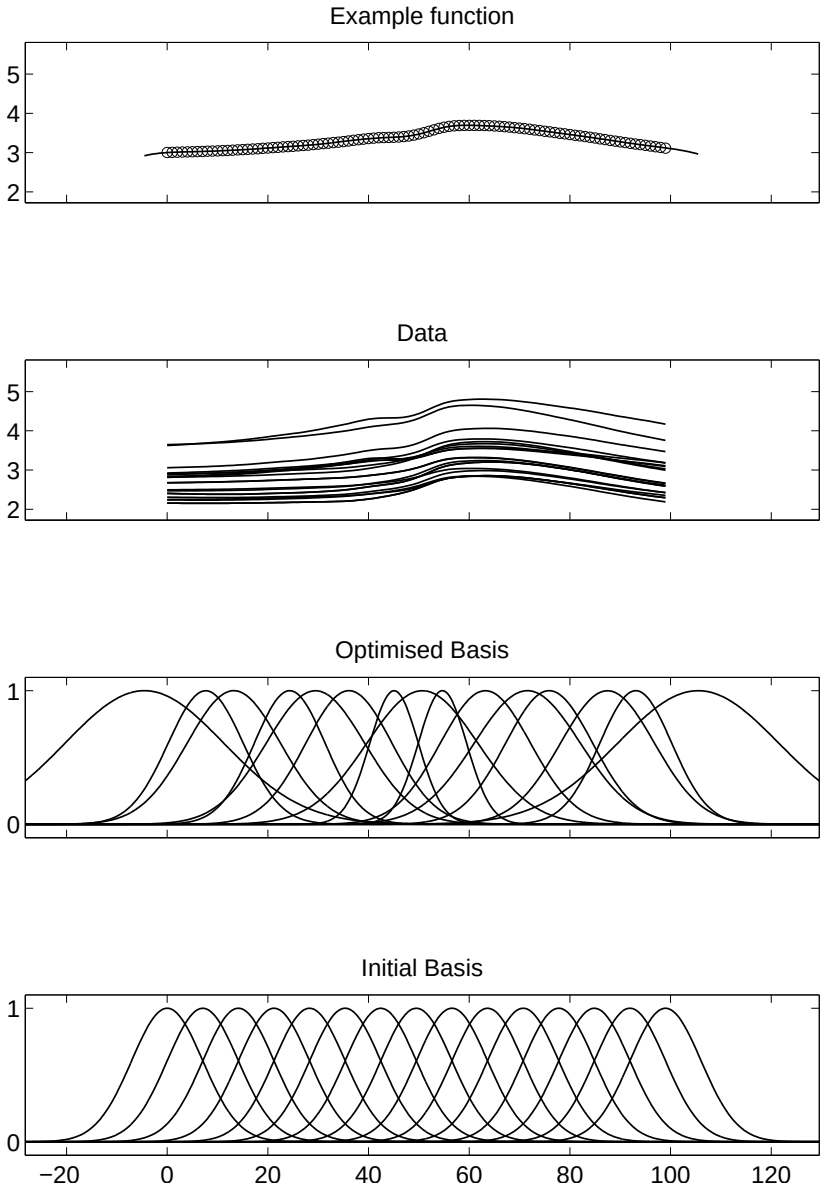

Figure A.1: Tecator: Optimised basis.

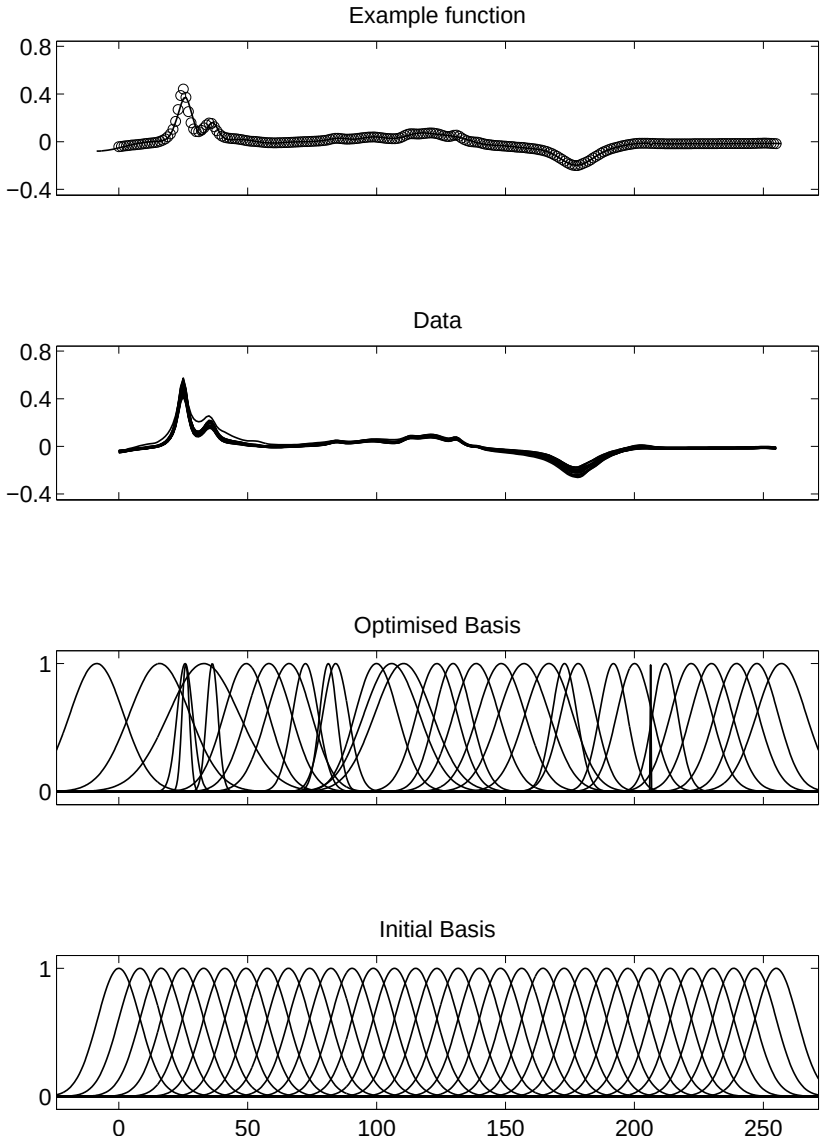

Figure A.2: Wine: Optimised basis.

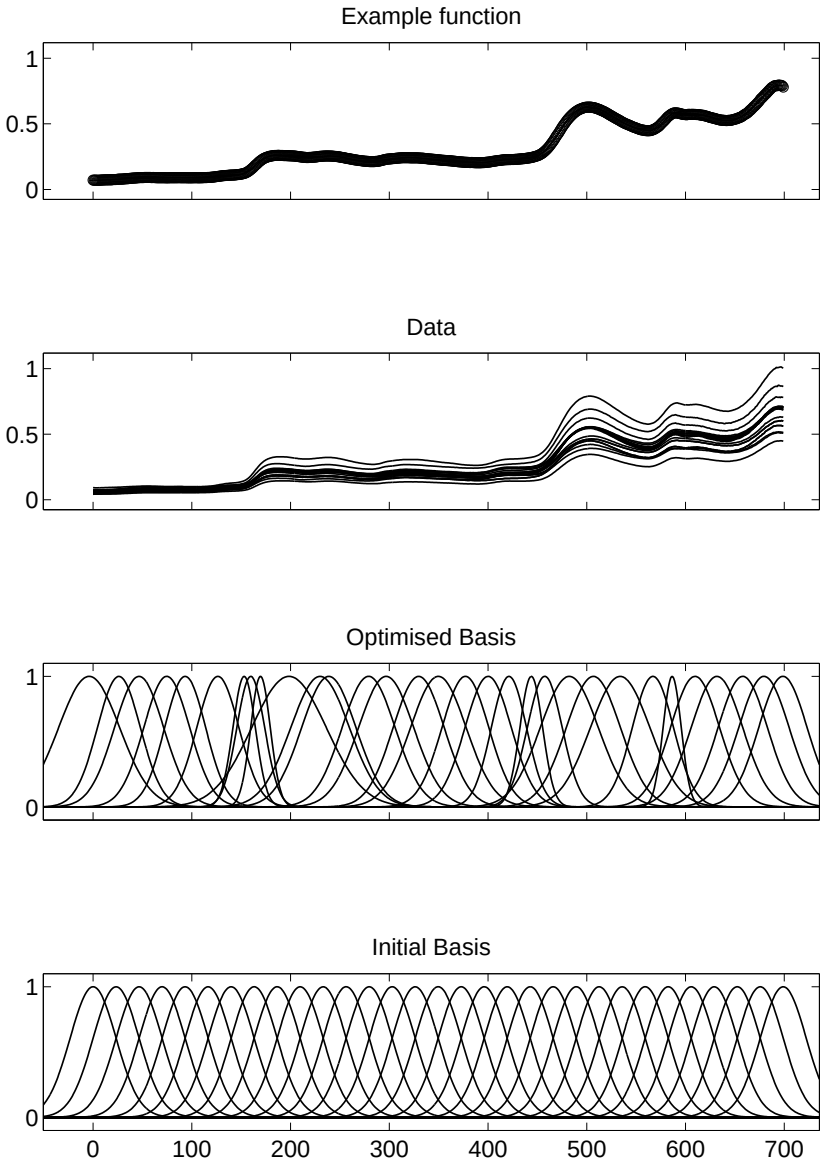

Figure A.3: Juice: Optimised basis.

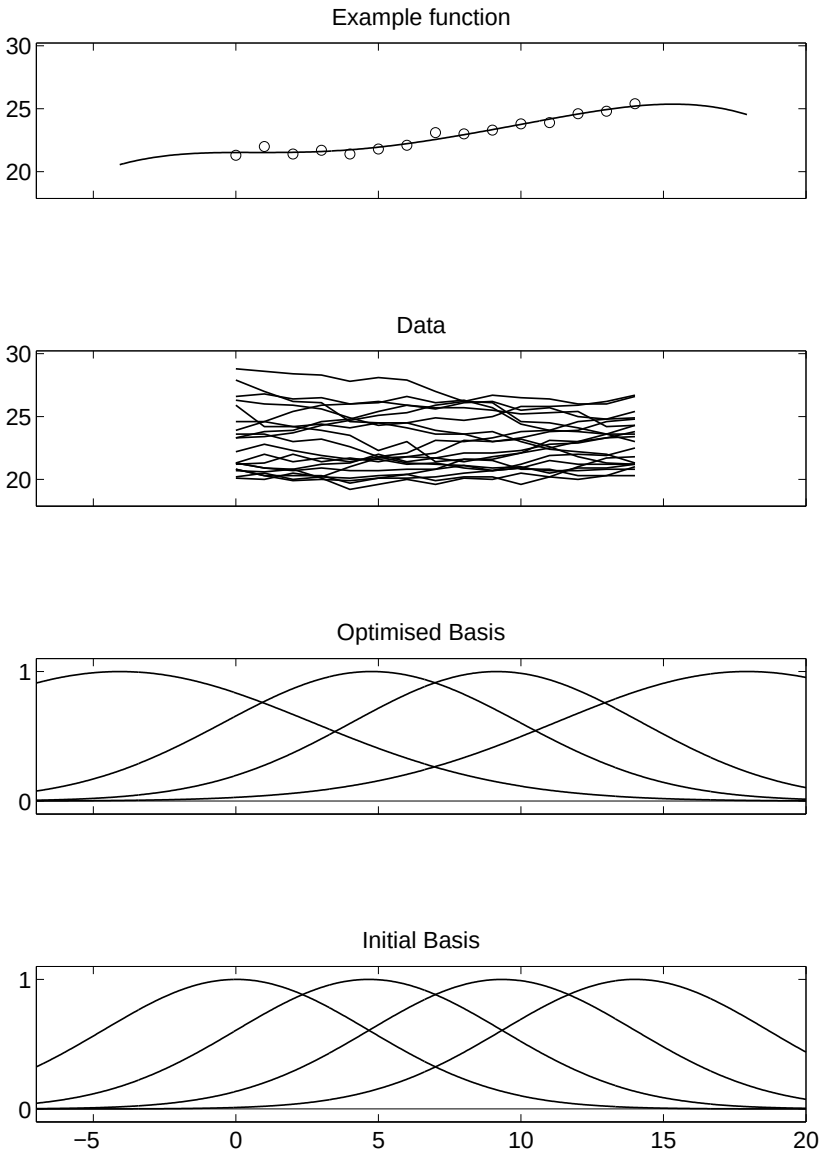

Figure A.4: ESTSP 15 steps ahead: Optimised basis.

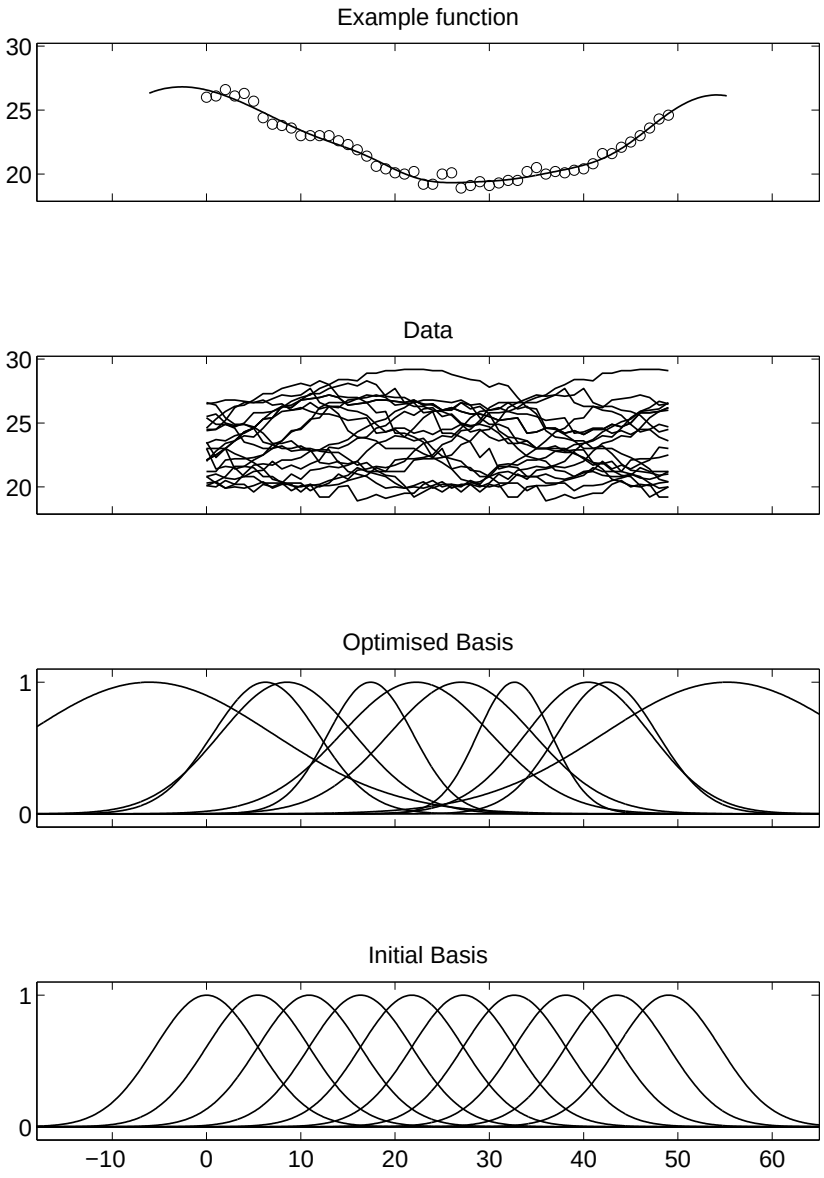

Figure A.5: ESTSP 50 steps ahead: Optimised basis.

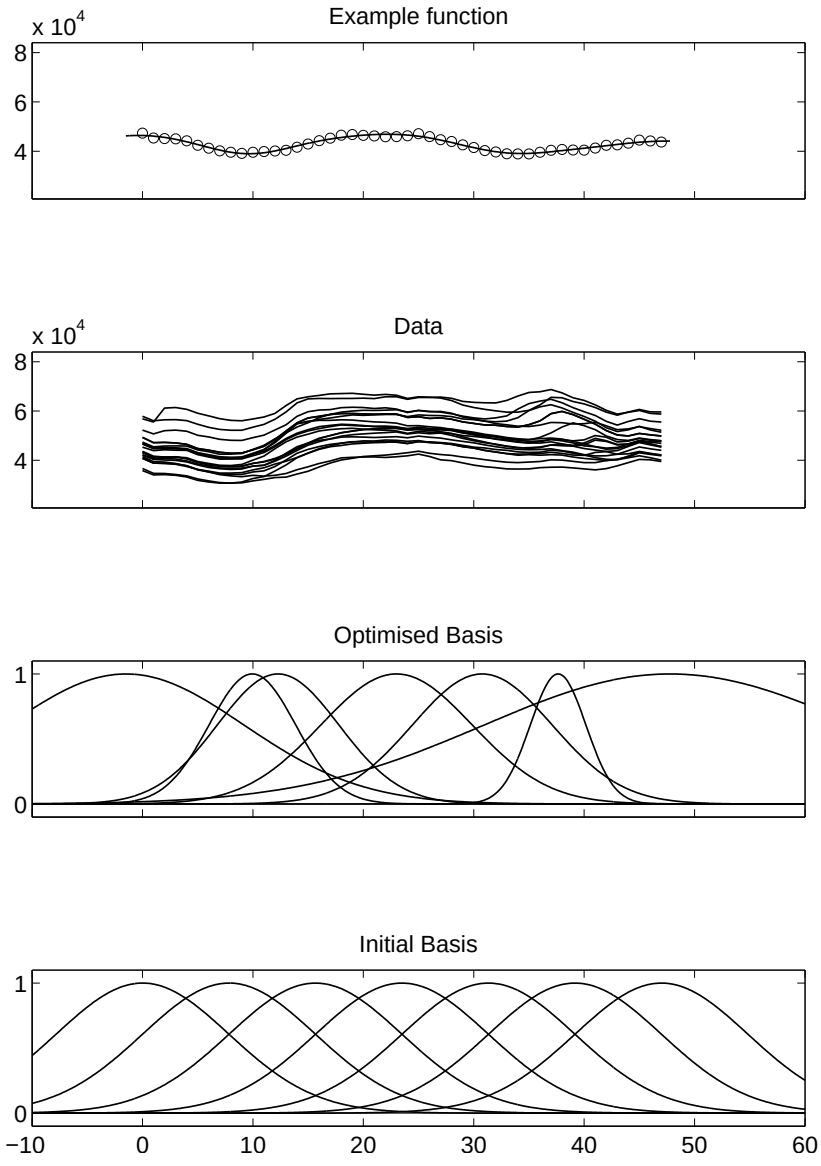

Figure A.6: RTE 48 steps ahead: Optimised basis.

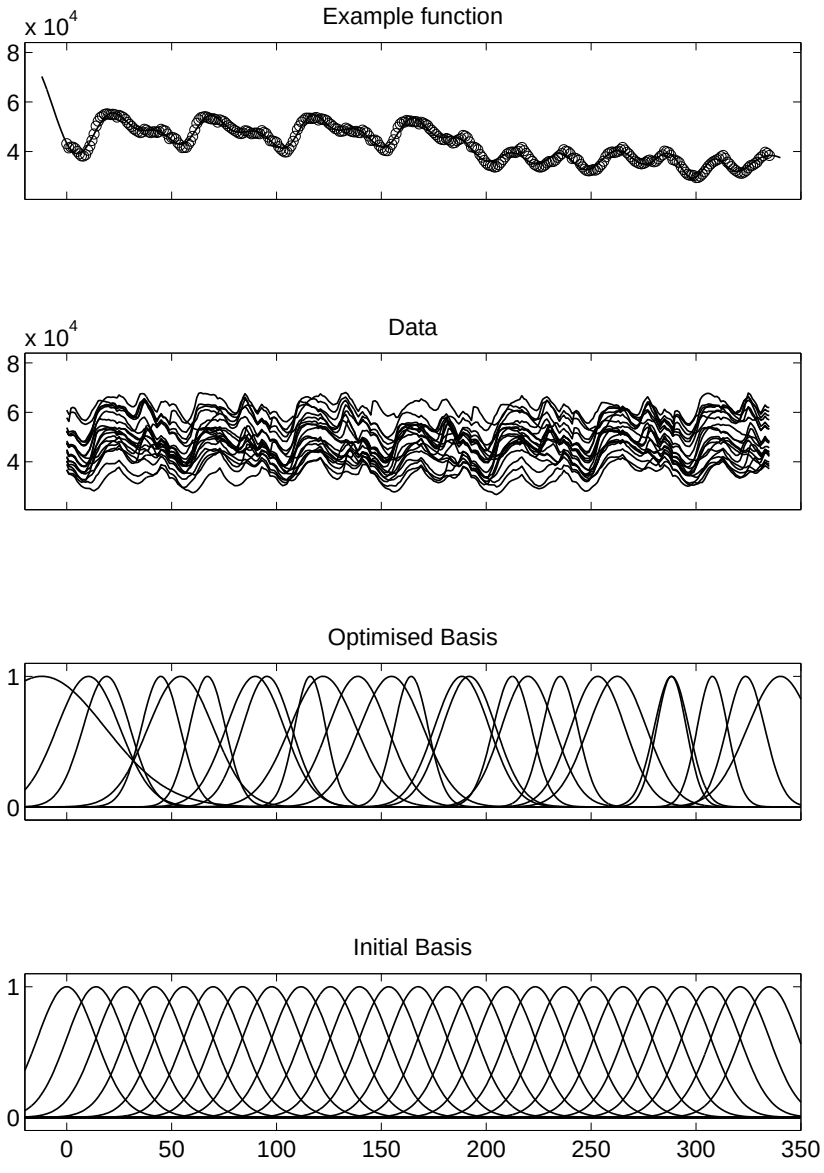

Figure A.7: RTE 336 steps ahead: Optimised basis.